



ISA

(Modules 1 to 6)

Background Material

INFORMATION SYSTEMS AUDIT 3.0 COURSE

Module - 3

**System Development, Acquisition,
Implementation and Maintenance
Application System Audit**



Digital Accounting and Assurance Board
The Institute of Chartered Accountants of India
(Set up by an Act of Parliament)
New Delhi

Background Material on Information Systems Audit 3.0 Course

Module-3 : System Development, Acquisition, Implementation and Maintenance Application System Audit



Digital Accounting and Assurance Board
The Institute of Chartered Accountants of India
(Set up by an Act of Parliament)
New Delhi

© The Institute of Chartered Accountants of India

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form, or by any means, electronic mechanical, photocopying, recording, or otherwise, without prior permission, in writing, from the publisher.

DISCLAIMER

The views expressed in this material are those of author(s). The Institute of Chartered Accountants of India (ICAI) may not necessarily subscribe to the views expressed by the author(s).

The information in this material has been contributed by various authors based on their expertise and research. While every effort have been made to keep the information cited in this material error free, the Institute or its officers do not take the responsibility for any typographical or clerical error which may have crept in while compiling the information provided in this material. There are no warranties/claims for ready use of this material as this material is for educational purpose. The information provided in this material are subject to changes in technology, business and regulatory environment. Hence, members are advised to apply this using professional judgement. Please visit PQC portal for the latest updates. All copyrights are acknowledged. Use of specific hardware/software in the material is not an endorsement by ICAI.

Revised Edition : August, 2020

Committee/Department : Digital Accounting and Assurance Board

Email : gdaab@icai.in

Website : www.icai.org/ <https://pqc.icai.org>

Price : ₹ 750/- (For Complete Set)

ISBN : 978-81-8441-995-5

Published by : The Publication Directorate on behalf of
The Institute of Chartered Accountants of India
ICAI Bhawan, Post Box No. 7100,
Indraprastha Marg, New Delhi - 110002

Printed by : Sahitya Bhawan Publications,
Hospital Road, Agra – 282 003
August | 2020 | P2724 (Revised)

Foreword

The digital revolution is transforming the traditional ways of doing business, necessitating realignment of profession to leverage the multipliers of digital technology - enhanced efficiency, scale and speed, effectiveness, agility and giving access to newer markets. In view of the rapid technological changes, it is imperative for Information System Auditors to adapt, be innovative in aiding organizations to improve its control environment and strengthen governance of IT risks. Adoption of emerging technologies will help them to assimilate vast amount of data and provide value added analysis in the form of data analysis and business intelligence. Chartered Accountants possess unique blend of systems and process understanding and expertise in controls and governance, thereby best suited to be the perfect Information Systems Auditor.

The Institute of Chartered Accountants of India (ICAI), through its Digital Accounting and Assurance Board (DAAB), is continuously monitoring technological developments and taking initiatives to disseminate updated knowledge amongst our members and other stakeholders. In this direction, it is heartening to note that the DAAB is bringing out next version of "Educational Material" for Post Qualification Course on Information Systems Audit. This updated and revised Material combines technology, information assurance and information management expertise that enable Chartered Accountants to be an advisor and handling assurance assignments.

In this updated course curriculum various aspects of emerging technologies like, Blockchain, Robotics Process Automation, etc., have also been introduced to keep members fully abreast. With focus on increased practical aspects, case studies and lab manuals at appropriate places this material is a great learning guide for members aspiring to be Information Systems Auditor.

I compliment CA. Manu Agrawal, Chairman, CA. Dayaniwas Sharma, Vice-Chairman and other members of the Digital Accounting and Assurance Board for generation next material in digital era by taking up this timely initiative.

I am confident that our members would take benefit of these updated modules of post qualification course on Information Systems Audit, so as to render their professional responsibility as Information System Auditor more efficiently and highest standards to achieve global recognition.

CA. Atul Kumar Gupta
President, ICAI

Place: New Delhi

Date: April 12, 2020

Preface

Evolution of digital economy and ever-changing dynamic ecosystem presents significant challenges, including new competition, new business and service delivery models, unprecedented transparency, privacy concerns and cyber threats. With a goal to keep members abreast of impact of emerging technologies, Digital Accounting and Assurance Board has come out with the updated Post Qualification Course on Information Systems Audit Modules to equip members with specialised body of knowledge and skill sets so that they become Information Systems Auditors (ISAs) who are technologically adept and are able to utilize and leverage technology to provide reasonable assurance that an organization safeguards its data processing assets, maintains data integrity and achieves system effectiveness and efficiency. This updated syllabus facilitates high level understanding about the role and competence of an IS Auditor to analyse, review, evaluate and provide recommendations on identified control weaknesses in diverse areas of information systems deployment.

Revised Modules of Post Qualification Course on Information Systems Audit has specific objective, i.e., “To provide relevant practical knowledge and develop skills for planning and performing various types of assurance or consulting assignments in the areas of Governance, Risk management, Security, Controls and Compliance of Information Systems.” The core of DISA 3.0 lies in inculcating competence to add to service delivery of the members. The updated course would help the members to apply appropriate strategy, approach, methodology and techniques for auditing information system and perform IS Assurance and consulting assignments by using relevant best practices, IS Audit standards, frameworks, guidelines and procedures.

The updated ISA Course 3.0 has a blend of training and includes e-learning, live case studies and lab manuals, project work in addition to class room lectures. This updated background material also includes a DVD which has e-Learning lectures, PPTs, case studies, DEMO CAAT software, useful checklists and sample audit reports. New Module on “Emerging Technology and Audit” has been added which covers Information System Assurance and Data Analytics, Assurance in Block chain Ecosystem, and Embracing Robotic Process Automation in Assurance Services. In addition to this Artificial Intelligence and Internet of Things (IoT) has also been inducted in the new modules.

We would like to take this opportunity to place on record our deep appreciation for the efforts put in by Convener, Dr. Onkar Nath as well as authors and reviewers of the various modules, viz., CA Anand Prakash Jangid, Mr. N.D. Kundu, Mr. Inder Pal Singh, Mr. Avinash Gokhale, CA Pranay Kochar, CA Naresh Gandhi, Dr. Manish Kumar Srivastava, Dr. Saurabh Maheshwari, CA Narasimhan Elangovan and CA Atul Kumar Gupta. It would be also appropriate to express our thanks to all the ISA faculties for giving their inputs/ suggestions for the implementation of DISA 3.0.

We would like to express gratitude to CA. Atul Kumar Gupta, President, ICAI, and CA. Nihar Niranjan Jambusaria, Vice President, ICAI, for their thought leadership and encouragement to the initiatives of the Board. We would also like to place on record our gratitude for all the Board members, co-opted members and special invitees for providing their valuable guidance and support in this initiative of the Board. We also wish to express my sincere appreciation for CA. Amit Gupta, Secretary, DAAB, Ms. Nishi Saraf, Section Officer for their untiring efforts in finalization of the updated Modules.

We are sure that these updated Modules on Post Qualification Course on Information Systems Audit would be of immense help to the members and enable them to enhance service delivery not only in compliance, consulting and assurance of IT services, but also provide new professional avenues in the areas of IT Governance, Cyber Security, Information System Control and assurance services.

CA. Manu Agrawal
Chairman
Digital Accounting and Assurance Board

CA. Dayaniwas Sharma
Vice-Chairman
Digital Accounting and Assurance Board

Contents

Learning Objectives	xi
Chapter 1: Project Management for SDLC	1
Objectives	1
1.1 Introduction	1
1.2 Project Management Frameworks	1
1.2.1 Capability Maturity Model Integration (CMMI)	2
1.3 Key concepts of Project Management	3
1.4 Program and Project Management and Organization	4
1.4.1 Portfolio/Program Management	4
1.4.2 Program/Project Management Organization Forms	5
1.5 Project Initiation	6
1.5.1 Project Management Methodology	7
1.5.2 Project Context and Environment	8
1.5.3 Project Communication and Culture	8
1.5.4 Project Objectives	9
1.5.5 Project Management Practices	9
1.6 Project Planning	10
1.7 Project Controlling	11
1.7.1 Management of Scope	11
1.7.2 Resource Management	12
1.7.3 Project Risk Management Standards and Methods	13
1.8 Project Closing	14
1.9 Roles and Responsibilities	15
1.9.1 Steering Committee	15
1.9.2 Project Sponsor	16
1.9.3 Project Manager	16
1.9.4 Senior Management	17

1.9.5	Business Management	17
1.9.6	Systems Development Project Team	17
1.9.7	Business Function Representatives/Domain Specialists	17
1.9.8	Security Officer	18
1.9.9	Quality Assurance (QA)	18
1.9.10	Technology Specialist	18
1.9.11	Systems Analyst	19
1.9.12	Programmers/Developers	19
1.9.13	Testers	19
1.9.14	Documentation Specialist	19
1.9.15	Database Administrator (DBA)	19
1.9.16	Data Administrator (DA)	19
1.9.17	User Manager	19
1.9.18	IS Auditor	19
1.10	SDLC Project Management Techniques and Tools	20
1.11	Summary	27
1.12	Questions	28
1.13	Answers and Explanations	30
Chapter 2	: SDLC – Need, Benefits and Phases	32
	Learning Objectives	32
2.1	What is SDLC?	32
2.2	Relevance of SDLC for Business Process Automation	32
2.3	Need for SDLC	33
2.4	Benefits of SDLC	33
2.5	Phases of SDLC	34
2.5.1	Feasibility Study	34
2.5.2	Requirement Definition	35
2.5.3	System Analysis	35

2.5.4	System Design	36
2.5.5	Development	37
2.5.6	Testing	39
2.5.7	Implementation	41
2.5.8	Maintenance	42
2.6	Types of SDLC Model	43
2.6.1	Waterfall Model	43
2.6.2	Incremental Model	45
2.6.3	Software Reengineering and Reverse Engineering	47
2.6.4	Object Oriented Software Development	49
2.6.5	Component Based Development	51
2.6.6	Web Based Application Development	52
2.7	Selection of SDLC Model	53
2.8	New Development Iterative Models- Prototype, Spiral, Rapid & Agile etc.	54
2.8.1	Prototype Methodology	54
2.8.2	Spiral Model	57
2.8.3	Rapid Application Development	59
2.8.4	Agile Software Development Methodology	61
2.8.5	DevOps	64
2.8.6	DevSecOps	64
2.9	Secure SDLC	65
2.10	Summary	66
2.11	Questions	67
2.12	Answers and Explanations	69
Chapter 3 Software Testing and Implementation		71
Learning Objectives		71
3.1	Introduction	71
3.2	Importance of Software Testing	71

3.3	Methods of Software Testing	72
3.3.1	Black Box Testing	72
3.3.2	White Box Testing	73
3.3.3	Grey Box Testing	73
3.3.4	A Comparison of Testing Methods	74
3.4	Levels of Testing	75
3.4.1	Functional Testing	75
3.4.2	Non-Functional Testing	76
3.5	Strategies of Software Testing	76
3.5.1	What is the Test Strategy	76
3.5.2	Different Test Approaches	76
3.5.3	Factors to be Considered	77
3.6	Types of Software Testing	77
3.6.1	Unit Testing	77
3.6.2	Static Testing	78
3.6.3	Load Testing	79
3.6.4	Usability Testing	79
3.6.5	Portability Testing	79
3.6.6	Integration Testing	80
3.6.7	Regression Testing	81
3.6.8	System Testing	81
3.6.9	Other Types of Testing	82
3.7	Final Testing	83
3.7.1	Quality Assurance Testing	84
3.7.2	User Acceptance Testing	84
3.8	Implementation	85
3.8.1	Implementation Strategies	85
3.8.2	Preparing for Implementation	87
3.8.3	Conversion	88

3.9	Change Management Process	90
3.9.1	Emergency Change	92
3.9.2	Implementing Changes to Production	93
3.9.3	Segregation of Duties	93
3.9.4	Configuration Management	94
3.10	Summary	95
3.11	Questions	95
3.12	Answers and Explanations	97
Chapter 4	Application Controls	100
	Learning Objectives	100
4.1	Introduction	100
4.2	What is Application Control?	100
4.2.1	Features and Benefits of Application Controls	101
4.3	Types of Application Controls	102
4.3.1	Input Controls	102
4.3.2	Processing Controls	103
4.3.3	Output Controls	104
4.3.4	Business Process Control Assurance	104
4.4	Application Controls Objectives	105
4.5	Design and Implementation of Application Controls	106
4.6	Application Controls and the System Development Life Cycle	107
4.7	Business Processes and Application Controls	108
4.7.1	Business Risk and Information Processing	109
4.8	Application Controls Assurance	109
4.8.1	Assurance over Application Controls	110
4.9	Summary	112
4.10	Questions	113
4.11	Answers and Explanations	114

Learning Objectives

- Evaluate whether proposed changes to information system are meeting business objectives.
- Evaluate policies and practices about the organization's project management.
- Evaluate the effectiveness of controls at all stages of SDLC
- Evaluate the process for migration of new system to the production
- Post implementation review of system to ensure that new system met business requirement, controls and project deliverables.
- Evaluate change management, configuration management, release management and patch management.

Chapter 1

Project Management for SDLC

Learning Objectives

This chapter provides insights on Project management aspect of System Development. This includes initiation of program/project, establishing project management methodology, defining objective, project risk management, planning, resource management, monitoring and controlling project, managing changes, closing project and tools and techniques required for software project management.

1.1 Introduction

In this chapter basic understanding about project management has been given. Unless the proposed system becomes operational and organization begins deriving benefit out of it, SDLC project cannot be treated as complete.

Control aspect of the proposed information system is planned at the design stage. IS Auditor should ensure that appropriate controls are designed at analysis and design stage.

1.2 Project Management Frameworks

Project is initiated once it is approved. In order to ensure that proposed project is successful, i.e. it meets its predefined objectives and delivers value, the organization must adopt effective and efficient project management practices. Without project management practices, tools and control frameworks, it is not possible to manage all the relevant aspects like planning, scheduling, resource management, risk management, sizing and estimation of efforts, milestone achievements, quality, deliverables and budget monitoring, of a large project. IS Auditor must understand the need for a project management framework within the organization, and associated elements required to establish a standard methodology. This chapter covers the various project management practices and how these are executed within the organization.

There are many approaches for project management defined by various professional bodies. The most commonly used approaches are:

- Project Management Body of Knowledge (PMBOK®) version 6, i.e. IEEE standard 1490 from the Project Management Institute (PMI),
- Projects in a Controlled Environment (PRINCE2™) from the Office of Government Commerce (OGC) in the UK, and the International Project Management Association (IPMA).

Since there are significant differences in scope, content and wording in each of these standards, an auditor has to be familiar with the standard adopted by auditee organization, prior to involvement in project. Although each project management approach has its own pros and cons, several elements are common across all project management methodologies. Some are focused on software development, others have a more general approach; some concentrate on a holistic and systemic view, others provide a very detailed workflow including templates for document creation

1.2.1 Capability Maturity Model Integration (CMMI)

Capability Maturity Model Integration (CMMI) is a process improvement approach that provides enterprise with the essential elements of effective processes. It can be used to guide process improvement across a project, division or entire organizational CMMI helps integrate traditionally separate organizational functions, set process improvement goals and priorities, provide guidance for quality processes and a point of reference for appraising current processes.

Table 1.1 – CMMI Levels

CMMI Levels	Description of Capability maturity levels	Context
Level 5 Optimized	The previously described predictable process is continuously improved to meet relevant current and projected business goals	Enterprise view/ Corporate knowledge
Level 4 Predictable	The previously described established process now operates within defined limits to achieve its process outcomes	
Level 3 Established	The previously described managed process is now implemented using a defined process that is capable of achieving its process outcomes	
Level 2 Managed	The previously described performed process is now implemented in a managed fashion (planned, monitored and adjusted) and its work products are appropriately established, controlled and maintained	Instance view/ Individual knowledge
Level 1 Performed	The implemented process achieves its process purpose	
Level 0 Incomplete	The process is not implemented or fails to achieve its process purpose. At this level, there is little or no evidence of any systematic achievement of the process purpose	

1.3 Key Concepts of Project Management

- Project is a temporary activity undertaken to generate defined outcome (like creating a service or product). Temporary need not be short, what it means is it has predefined beginning and end. For example, a project can be initiated to build a housing complex, that is completed once tenants have occupied it, or it can be for building infrastructure, or designing a new product, e.g. Tata motors designing the Nano car. The project is closed once the expected outcome is delivered or results are achieved or if the project becomes technically or economically unviable. In short projects can be initiated for any reason. Developing Software and deploying it can be a project or depending upon size, it can be group of projects.
- A project management refers to the practice of managing a project. Management may not want to initiate a project as it involves providing resources and waiting till the end to get deliverables. Management has to monitor the progress of project and intervene periodically to ensure that the project finally achieves the defined objectives. Project management practice is a set of multiple processes grouped in five major process groups:
 - Project Initiation
 - Project Planning
 - Project Execution
 - Project Controlling and Monitoring
 - Project Closing

Although these processes are grouped, they are not executed in sequence, Process groups under project planning, project execution and controlling are executed in iteration. (Figure 1.1)

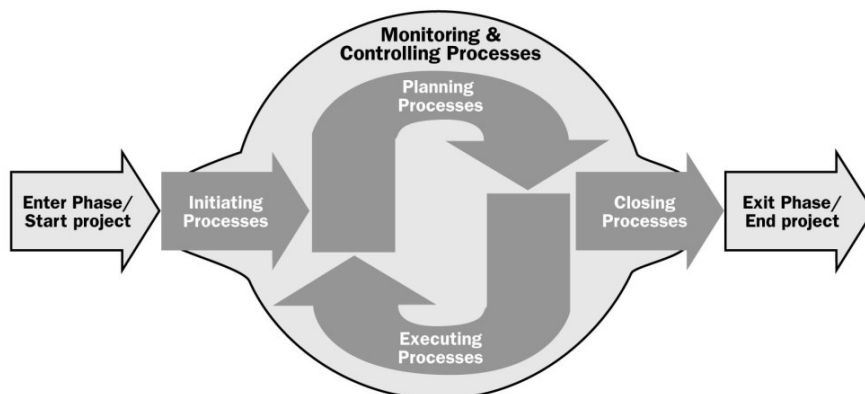


Figure 1.1: Project process groups

Each group mainly consists of processes; however, all processes may not be applicable to all projects.

Project initiation group consists of mainly processes related to developing project charter based on scope of project. In SDLC project, it is business case that help in Identifying beneficiaries and stakeholders of project.

Project planning consists of processes related to developing project execution plan, finalizing requirements, defining work breakdown structure and modules to be developed, estimating efforts and cost, resource planning, risk management, procurement planning and plan for communications with stakeholders.

Project execution consists of processes related to direct project teams, ensuring quality assurance and testing, managing requirements and changes in requirements, ensuring timely procurements and manage resources.

Project monitoring and controlling consists of processes related to monitoring risks, Scope Creeps, quality of deliverables, costs and budgets, performance reporting.

Project closing has processes for handing over deliverables or terminating project. SDLC project management is further discussed in ensuing sections.

1.4 Program and Project Management and Organization

1.4.1 Portfolio/Program Management

A program is a group of projects and/or time-bound tasks that are linked together through common objectives. It may share a common budget and can have intertwined schedules. Like projects, programs have a limited time frame (start and end date), predetermined budget, defined deliverables/outcomes and at many times organizational boundaries. A program is more complex than a project and many times consists of multiple projects. For example, implementing ERP at all plants can be a program consisting of multiple projects for implementing ERP at each plant. (Figure 3.2 explains the relationship of portfolio, programs and projects)

A **portfolio** is group of all projects/programs (related or unrelated) being carried out in an organization at a given point in time.

A **project/program management office** (popularly referred as PMO) controls and manages Portfolios, programs and projects. PMO also governs the processes of project management but not involved in management of project content.

The program management includes management of:

- Program scope
- Program financials (costs, resources, cash flow, etc.)

- Program schedules
- Program objectives and deliverables
- Program context and environment
- Program communication and culture
- Program organization

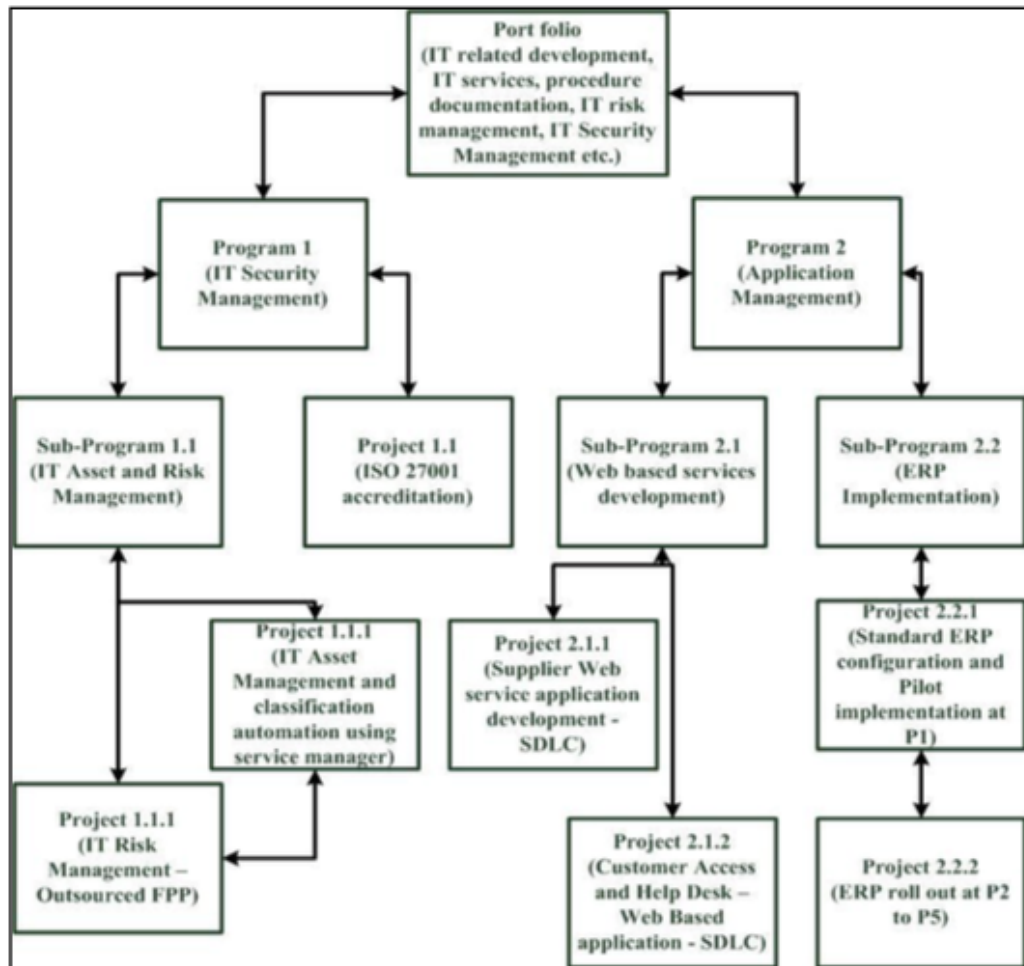


Figure 1.2: Portfolio, program and project

1.4.2 Program/Project management Organization Forms

A project may be considered as a group of complex tasks executed by a temporary

organization. However, depending upon the nature of business, from a project management perspective, organizations can be categorized as follows:

- **Functional organization that is influenced by the projects:** These are business organizations that are involved in production of goods and services. Projects are undertaken to support the functional activities. For example, a manufacturing organization may want to automate administrative processes (like finance, HR, pay roll etc.) using IT. In such organizations, Project Manager has only a staff function without formal management authority. The Project Manager is only allowed to advise peers and team members as to which activities should be completed. In such organization project team consist of staff that report to functional manager, except for the purpose of project activities assigned, reports to Project Manager.
- **Projectile organization:** These are pure project organizations that execute projects. For example, an infrastructure development organization or consulting organizations that executes projects. In such organizations Project Manager has formal authority over those taking part in the project. Often, this is bolstered by providing a special working area for the project team that is separated from their normal office space.
- **Matrix project organization:** The organization that provides product and services and also executes projects. Most IT companies falls under such categories where these organizations undertake project to manage business functions for other organizations and also executes projects for customer organization. In such organizations, Management Authority is shared between the Project Manager and the Department Heads.

IS Auditor has to understand these organizational forms and their implications on controls in SDLC project management activities.

1.5 Project Initiation

Whenever a business entity decides (i.e. stakeholders in the business or senior management) to undertake computerization, a project will have to be initiated. Some examples of a formal project initiation are:

1. A new business application is required to be developed to address a new or existing business process e.g. HR management system, billing system, order processing etc.
2. Adoption of a new technology invented or available becomes advantageous to the business e.g. Internet based advertising for an advertising company.
3. The application software to be developed, is expected to rectify the present problem related to existing business e.g. computerization of college admissions.
4. The application software to be developed, is expected to rectify the present problem related to existing technology e.g. migrating from text-based computerized system to

GUI based system as in case of old COBOL / XBASE based distributed banking to RDBMS based Core Banking system.

A project may be initiated from any part of the organization, including the IS department. A project is time bound, with specific start and end dates, a specific objective and a set of predetermined deliverables. Once a project is initiated, a project sponsor and project manager is appointed to execute the further activities. This also includes gathering information related to gaining approvals for the project. This will often be compiled into terms of reference or a project charter that states the objective of the project, the stakeholders in the system to be produced, and the Project Manager and sponsor. Approval of a project initiation or project request is authorization for a project to begin.

During the project initiation phase, several activities are performed by Project Manager starting from assessing the size, scope, as well as project complexity, and further establishes procedures to supporting subsequent activities. IS Auditor has to understand the implications on controls in SDLC project management activities. The major activities to be performed in the project initiation are:

- Establishment of project initiation team: In this activity an initial core of project team members are organized to complete the project initiation activities.
- Establishment of relationship with customer: A good understanding of the customer is needed to build stronger customer partnerships and also higher trust level.
- Establishment of plan for project initiation: This step provides the definition of activities required to organize the initiation team, who define the scope of the project.
- Establishment of management procedures: Without developing effective management procedure, it not possible to achieve successful completion of project.
- Establishment of project workbook and project management environment: The objective of this activity is to organize and collect the tools that will be used for managing the project and will help to develop the project workbook. For example, major portion of the project workbook is derived from charts, diagrams and description of the system. Thus, the project workbook serves as a repository for all project deliverables, inputs, outputs, correspondence, procedures, and standards established by the project team.

Many organizations that follow standard process for project management prepare a formal Project Initiation Report that is presented to Senior Management or Board of Directors. Once accepted this becomes formal charter for the project and triggers next phases of SDLC.

1.5.1 Project Management Methodology

IT projects are divisible into pre-defined phases. The project management process begins with the project charter and ends with the closure of the project. Since the project management can be complex, like other business processes, it also requires a standardized approach.

Organizations may adopt standard processes prescribed by globally accepted standards developed by organizations like PMI or can define a project management process within organization based on such prescribed standards. Organizations following a standard project management process have higher possibility of completing projects in time, within budget and deliverables meeting with expected quality. The following section explains general project management practices used by various organizations.

1.5.2 Project Context and Environment

Organization may be running several projects at the same time. These projects need not be SDLC projects or IT projects. At organization level the relationships between these projects have to be established to identify common objectives for the business, which is a function of a project portfolio management and/or a program management. This helps in consolidating common activities (e.g. identify and manage risks) and managing of common resource requirements. (Refer Figure 1.1)

A context of the project may be determined based on:

- Importance of the project deliverables to organization's objectives
- Connection between the organization's strategy and the project outcome
- Relationship with other projects
- Priority based on the business case In addition while considering the time context of the project, following aspects must be considered:
- Start and end time of the project, particularly if it is expected that the outcome of project has linkages to other projects.

The objective is to determine whether all relevant environments for the project, which will have a significant influence on overall project planning and project success, have been considered.

1.5.3 Project Communication and Culture

Success of project depends upon timely communication with stakeholders and affected parties. This can be achieved by:

- One-on-one meetings
- Kick-off meetings
- Project start workshops
- Periodic reporting

Communication helps in obtaining cooperation from all team members and buy-in from stakeholders. One of the major activities for Project Manager during execution of project is to

develop and execute communication plan so as to inform issues, concerns, if any and to report project progress.

1.5.4 Project Objectives

Primary objective of project is to deliver the defined outcome/deliverables/product in time, within budget and of desired quality. The measurement of success depends upon clearly defining results that are specific, measurable, attainable, realistic and timely (SMART). The main objectives of project are always directly coupled with business expectations. Additional objectives are objectives that are not directly related to the main results of the project but may contribute to project success (e.g., business unit reorganization in a System Development project).

A commonly accepted approach to define project objectives is to start with a work breakdown structure (WBS) with each work module having its own objectives derived from main objectives. The WBS is a tool used for the project in terms of manageable and controllable units of work and forms the baseline for cost and resource planning. Detailed specifications regarding the WBS can be used to develop work packages (WP). Each WP must have a distinct owner and a list of main objectives, and may have a list of additional objectives. The WP specifications should include dependencies on other WPs and a definition of how to evaluate performance and goal achievement.

A task list is a list of actions to be carried to complete each work package and includes assigned responsibilities and deadlines. The task list aids the individual project team members in operational planning and scheduling, that when merged together forms a project schedule. Project schedules are work documents containing the start and finish dates, percentage completed, task dependencies, and resource names of individuals planned to work on tasks.

1.5.5 Project Management Practices

Every organization uses project/program to implement new concepts, changes, business strategies etc. Collective knowledge of executing projects may be used to execute the projects; however, it is a prudent for the organization to adopt a standard project management practice, across entire organization. Many organizations prefer to adopt the practices based on global standards/best practices e.g. PMBOK, Prince2 etc.

Project management is the application of knowledge, skills, tools and techniques to a broad range of activities to achieve organizational objectives. For example: meeting user requirements by developing/acquiring new software within budget and timelines. Project management practices consist of defined processes for initiating, planning, executing, controlling and closing a project.

A successful project planning is a risk-based management process that is iterative in nature. Project management practices for SDLC projects also provide standards for systematic

quantitative and qualitative approaches to software size estimating, scheduling, allocating resources and measuring productivity. There are numerous project management tools available (e.g. MS project) that can be adapted to implement techniques to assist the Project Manager in controlling the time and resources utilized during execution of project.

1.6 Project Planning

To plan and control SDLC projects, Project Manager needs to determine:

- The various project tasks and management tasks that need to be performed to develop/acquire and implement business application system.
- The order in which these tasks should be performed.
- The estimated duration for each task.
- The priority of each task.
- The IT resources, available, transferred/loaned or to be acquired to perform these tasks.
- Budget or costing for each of these tasks. This can be notional for internal resources and ^{1.1.1}monetary for outsourced projects.

In complex projects the planning is dynamic and has to be reviewed/adjusted at the beginning and end of each project phase. This is to ensure that resources are available, quality of work during earlier phase has been as expected (i.e. no rework is required, or if required adjusting the plan by considering the delay and so on.)

There are some techniques like Gantt chart, Program Evaluation Review Technique (PERT), Critical Path Method (CPM) etc., that are useful in creating and monitoring project plan. The major activities which are performed during project planning are:

- Measure the development efforts. (Different software sizing techniques are discussed in section 1.8.)
- Another activity is to identify resources (e.g., people with requisite skills, development tools, facilities) for high level software development.
- Budgeting is next activity. Although overall budget for the project has been allocated at high- level during business case development, Project Manager need to prepare granular budget for monitoring. This is done by considering the cost for each resource and their expected use. For example, group of testing professionals might be required in the project, however they need not be available from the beginning of project and thus can be inducted (on boarded) at a later date thus optimizing the cost associated with their release from another project.
- Scheduling and establishing the time frame is another activity. While budgeting involves adding up the cost for human and machine resource usage involved in each task,

scheduling involves establishing when these resources are required in the project. This is achieved by arranging tasks according to:

1. The logical sequential and parallel tasks relationship and determining earliest start date.
2. Based on estimated efforts (section 1.7) for each resource arriving at latest expected finish date.
3. Schedules are presented using PERT, CPM diagrams and Gantt Charts. (Discussed in section 1.7.)

1.7 Project Controlling

The controlling activities of a project include:

- Management of Scope
- Monitoring of Resource Usage
- Risk Management.

It is critical to ensure that new requirements for the project are documented and, if approved, appropriate resources are allocated. Control of changes during a project ensures that projects are completed meeting stakeholder requirements of: time, use of funds and quality objectives. Stakeholder satisfaction should be addressed with effective and accurate requirements capture, proper documentation, baselining and skilled steering committee activity.

During mid-term project review IS Auditor should focus on project planning and controlling activities to ensure that these are not deviating from primary objectives of the project.

1.7.1 Management of Scope

Quite often, it is noticed that the majority of the SDLC projects suffer from "Scope Creep". This happens particularly when the requirement analysis is incomplete or the dynamic nature of business environment forces users to include these requirements and all these requirements cannot be put on hold. The Scope Creep affects the project planning seriously. This can be controlled by:

- Baselining the requirements before project planning.
- Establishing process for change management to decide which requirements must be included during development and how it is affecting the project time, cost, quality and outcome. Change management process must define who can request for change, how a formal change request be made, what it should contain and the reasons for the change. For complex deliverables, it is best to document the work breakdown structure.

- The Project Manager then assesses the impact of change request on project activities, schedule and budget.
- A change advisory board is appointed to evaluate change requests and decide on approving changes.
- If the change is accepted, the Project Manager should update the project plan.
- The updated project plan must be formally confirmed by the project sponsor—accepting or ^[1]_[SEP]rejecting the recommendation of the change advisory board.

1.7.2 Resource Management

Monitoring resource usage in project execution is the process to control budget and ensure that cost plan is on track. Budget and project plan assume certain productivity of resources. For example, if a program development is expected and hence planned to take 16 person-hours, then it is supposed that the resource being deployed is capable of finishing that task in 16 person-hours with expected quality level. (Using coding standards might help in improving productivity). Whether this is actually happening can be verified using Earned Value Analysis (EVA).

Earned Value Analysis consists of comparing expected budget till date, actual cost, estimated completion date and actual completion at regular intervals during the project. In above example the program development is expected to take two working days, with eight hours spent each day. At the end of first day cost is as per budget but EVA cannot be determined unless 50% or more work has been completed. The other alternative is to get information on how much time is required to complete remaining program. If the answer is 8 hours, the project is on track. If it is less, resource might be idle and if it is more, the project might be delayed. In short, at the end of first day, the resource spent is according to budget, but the “Earned Value” will be based on time remaining to complete the task. (Figure 1.3)

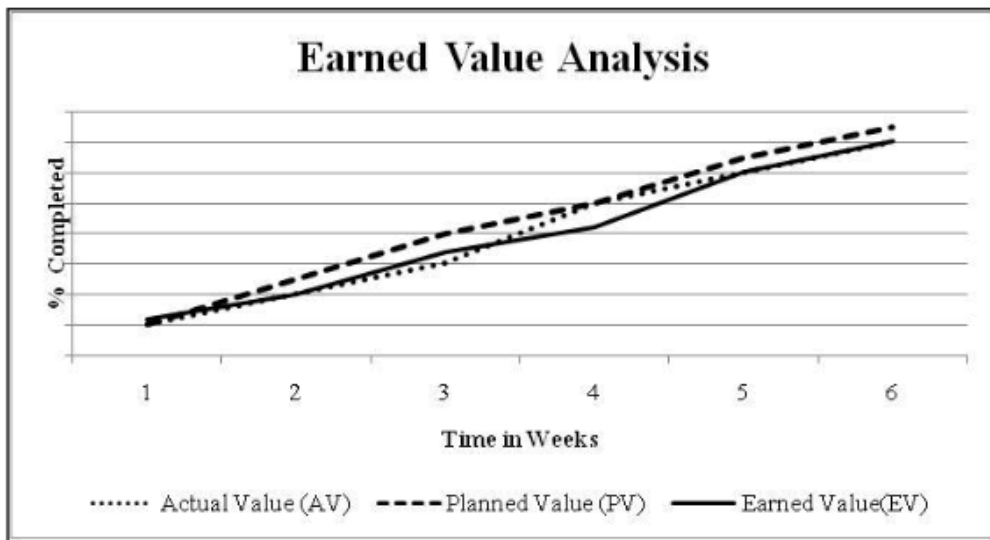


Figure 1.3: Earned Value Analysis^[1]

1.7.3 Project Risk Management Standards and Methods

Project Management practices require that the Project Manager invariably adopts the risk management frameworks. PMBOK of PMI specifies following activities for Project Manager:

Project Planning Phase

- Plan Risk
- Identify Risk,
- Qualitative Analyses of Risks
- Quantitative Analysis of Risks
- Plan Risk Response

Project Monitoring Phase

- Control Risks

Risk in Project Management

Risk is defined as a possible negative event or condition that would disrupt relevant aspects of the project. There are two main categories of project risk: the category that impacts the business benefits (and therefore endangers the reasons for the project's very existence) and the category that impacts the project itself. The project sponsor is responsible for mitigating the first category of risk and the Project Manager is responsible for mitigating the second


category. The risk management process consists of five steps that are repeatedly executed during a project. Phase-end milestones are a good anchor points in time at which the review and update of the initial risk assessments and related mitigations can be done.

Risk Management Process

- **Identify Risk:** Perform a brainstorming session with your team and create an inventory of possible risks.
- **Assess and Evaluate Risk:** Quantify the likelihood (expressed as a percentage) and the impact of the risk (expressed as an amount of money). The “insurance policy” (total impact) that needs to be in the project budget is calculated as the likelihood multiplied by the impact.
- **Manage Risk:** Create a Risk Management Plan, describing the strategy adopted and measures to deal with the risk. Generally, the more important the risk, the more budget should be made available for counter-measures. Counter-measures could include prevention, detection and damage control/reconstruction activities. Any risk can be mitigated, avoided, transferred or accepted depending on its severity, likelihood and cost of counter-measures and the organization’s policy.
- **Monitor risk:** Discover risk that materializes, and act accordingly.
- **Evaluate the Risk Management Process:** Review and evaluate the effectiveness and costs of the Risk Management Process.

IS Auditor has to focus on the Risk Management Process as it provides detailed insight on the effectiveness of Project Management.

1.8 Project Closing

Projects should be formally closed to provide accurate information on project results, improve future projects and allow an orderly release of project resources. The closure process should determine whether project objectives were met or excused, and should identify lessons learned to avoid mistakes and encourage repetition of good practices. Project closure is to be planned in two situations: 

1. Project deliverables are completed and are ready to be implemented:

1. The Project Sponsor should be satisfied that the system produced is acceptable and ready for implementation/delivery.
2. Custody of contracts may need to be assigned, and documentation archived or passed on to those who will need it.
3. Survey the project team, development team, users and other stakeholders to identify any lessons learned that can be applied to future projects.

4. Achievement of objectives of project and performance fulfilment, adherence to the schedule, costs, and quality of the project.
 5. Post project review in which lessons learned and an assessment of project management processes used are documented.
 6. Release of project teams either to other projects or line functions.
2. Project is suffering from Risk Materialization and has to be terminated.

These are generally exceptional situations like changes in functional requirements, obsolescence of planned technology, availability of new technology, unforeseen budget constraints, strategy changes etc. In rare cases the project may have to be terminated due to non-performance of project teams. In such situations closure of project may have to be planned depending upon the status of project. For example, based on project planning, organization may have placed order for required software and hardware or might have made some changes to its existing infrastructure. These need to be undone or planned so as to minimize the impact on organization.

IS Auditor conducting review after project closure has to consider the overall project execution on various parameters such as objectives achieved, time overrun, cost overrun, quality of deliverables, etc. If the review is being done immediately after implementation, IS Auditor may also review the challenges faced by the users and the resolution methods.

Achieving business objectives must be the focus of project review. Accordingly, the auditor may review and comment on budget and time overrun situations.

1.9 Roles and responsibilities

The various roles and responsibilities of groups/individuals that are associated with the project and program management for SDLC project are described below:

1.9.1 Steering Committee

Project Steering Committee provides overall direction and monitors the project execution. This is assured by representation of major stakeholders. The project steering committee is ultimately responsible for all deliverables, project costs and schedules.

This committee should comprise of senior representatives having authority for decision making, from business areas likely to be impacted by the proposed system or change. Mostly Project Sponsor will chair the steering committee. The Project Manager is a member of steering committee.

Role of Project Steering Committee Project Steering Committee performs the following functions:

1. Reviews project progress periodically (fortnightly or monthly or as required)
2. Serves as co-ordinator and advisor to the project. Members of the committee should be available to make user-related decisions about system and program design.
3. Takes corrective action based on reviews. The committee should evaluate progress and take action or make recommendations to resolve project issues related to budget, schedules, resources, and scope and project objectives.
4. Assess risks and decide upon mitigation plan. Also resolve issues that are escalated and cannot be resolved at the project level.
5. Take decision on and if required recommend the project be halted or discontinued.
6. Work closely with the Project Manager to define project success factors and metrics in measurable and quantifiable terms.

1.9.2 Project Sponsor

Head of Business Function or Senior Management (generally who has the highest stake in benefit realization from the project) is designated as Project Sponsor. Project Sponsor provides funding and assumes overall ownership and accountability of the project. Project Sponsor is also responsible for providing funding and budget for the project execution.

1.9.3 Project Manager

A Project Manager should be identified and appointed by the IS steering committee. The Project Manager, who need not be an IS staff member, should be given complete operational control over the project and be allocated the appropriate resources, including IS professionals and other staff from user departments, for the successful completion of the project. A Project Manager is appointed for execution of project. The Project Manager can be from the user department, or from IS department or hired separately to handle the project. Primary functions of Project Manager are:

1. Provide day-to-day management and leadership.
2. Ensure that project activities are in line with pre-determined objectives.
3. Involve affected departments.
4. Follow organization's Project Management Standards.
5. Ensure expected quality of deliverables.
6. Resolve conflicts.
7. Monitor and controls costs, schedules and associated risks.

1.9.4 Senior Management

Demonstrates commitment to the project and approves the necessary resources to complete the project. This commitment from senior management helps ensure involvement by those needed to complete the project. Generally senior management representative is appointed by the steering committee

1.9.5 Business Management

Business Management, most of the times, assumes ownership of the project and resulting system, allocates qualified representatives to the team, and actively participates in business process redesign, system requirements definition, test case development, acceptance testing and user training. Business Management should review and approve system deliverables as they are defined and implemented.

Business Management is concerned particularly with the following questions:

1. Are the required functions available in the software?
2. How reliable is the software?
3. How efficient is the software?
4. Is the software easy to use?
5. How easy is it to transfer or adapt old data from pre-existing software to this environment?
6. How easy is it to transfer the software to another environment?
7. Is it possible to add new functions?
8. Does it meet regulatory requirements?

1.9.6 Systems Development Project Team

System development team consist of System Analyst, Developers, Testing Professionals, Control Consultants (IS Auditor), Hardware and Network Consultants, The team members complete the assigned tasks, communicate effectively with users by actively involving them in the development process, work according to local standards and advises the Project Manager of necessary project plan deviations.

1.9.7 Business Function Representatives/Domain Specialists

Consists of Subject Matter Experts (SME) that provides inputs to developers and system analysts on requirements, business related controls, and sometime approves the low-level design specifications.

1.9.8 Security Officer

Ensures that system controls and supporting processes provide an effective level of protection, based on the data classification set in accordance with corporate security policies and procedures; consults throughout the life cycle on appropriate security measures that should be incorporated into the system; reviews security test plans and reports prior to implementation; evaluates security- related documents developed in reporting the system's security effectiveness for accreditation; and periodically monitors the security system's effectiveness during its operational life.

1.9.9 Quality assurance (QA)

Quality assurance function consists of following key activities:

- Develop test plan and test the code.
- Review and ensure that Project Documentation is complete.
- Review deliverables of the project.

The objective is to ensure that the quality of the project by measuring the adherence by the project staff to the organization's standard methodology of System Development Life Cycle (SDLC), advise on deviations, and propose recommendations for process improvements or greater control points when deviations occur.

Specific Objectives of the QA function include:

1. Ensuring the active and coordinated participation by all relevant parties in the revision, evaluation and dissemination, and application of standards, management guidelines and procedures
2. Ensuring compliance with the agreed-on systems development methodology
3. Reviewing and evaluating large system projects at development milestones, and making appropriate recommendations for improvement
4. Establishing, enhancing and maintaining a stable, controlled environment for the implementation of changes within the production software environment
5. Defining, establishing and maintaining a standard, consistent and well-defined testing methodology for applications
6. Reporting to management on systems that are not performing as defined or designed

1.9.10 Technology Specialist

IT is developing so rapidly that even IT professionals find it difficult to keep track of all developments, let alone develop expertise. This has resulted in experts in specific technology areas, such as Microsoft technology, Web-enablement and the like.

1.9.11 Systems Analyst

The System Analyst also has a responsibility to understand existing problem/system/data flow and new requirements. System Analysts convert the user's requirements in the system requirements to design new system.

1.9.12 Programmers/Developers

Programmers convert design into programs by coding using programming language. They are also referred to as Coders or Developers.

1.9.13 Testers

Testers are junior level quality assurance personnel attached to a project. They test programs and subprograms as per the plan given by the module / project leaders and prepare test reports.

1.9.14 Documentation Specialist

These professionals are responsible for the creation of user manuals and other documentation.

1.9.15 Database Administrator (DBA)

The data in a database environment has to be maintained by a specialist in Database Administration so as to support the application program. The Database Administrator handles multiple projects; and ensures the integrity and security of information stored in the database.

1.9.16 Data Administrator (DA)

Data administrator gathers and analyzes business requirements and develops conceptual and logical models of business. He/she defines and enforces standards and naming conventions of database. Management and administration of metadata repository and data administration tools are entrusted to Data Administrator. He/she also keeps interface with business users for data definition.

1.9.17 User Manager

User Manager is the immediate manager or reporting manager of an employee. They have ultimate responsibility for all user IDs and information assets owned by company employees. In the case of non-employee individuals such as contractors, consultants, etc., User Manager is responsible for the activity and for the company assets used by these individuals.

1.9.18 IS Auditor

The IS Auditor can be a part of SDLC project team as consultant for internal controls or for

the review of the project activities. They may also provide an independent, objective review to ensure appropriate level of commitment of the responsible parties. IS Auditor has to understand the systems development; acquisition and maintenance methodologies used by the organization and identify potential vulnerabilities. If auditor observes control weakness either as a result of review due to organizational structure or the software methods used, or weakness in process execution, it is the IS auditor's role to advise the project team and Senior Management of the deficiencies in project management and provide recommendations for improvement.

Role of IS Auditor in SDLC

Throughout the project management process, IS Auditor should analyze the associated risks and exposures inherent in each phase of SDLC. He should assure that appropriate control mechanisms are in place to minimize the risks in a cost-effective manner, while reviewing SDLC various phases as well as the project team meetings Minutes. He will also assess the project development team's ability to produce key deliverables by the promised dates. Adequate and complete documentation of all phases should be collected and reviewed by processes, IS Auditor is expected to obtain necessary and available documentation from the Project Manager. The specific areas of review are:

1. Understand standards adopted and followed by the organization through the process of inquiry, observation and documentation review.
2. To determine significant phases for the various size and type.
3. To assess efficiency and effectiveness of each function to satisfy the users goals and organization objectives.
4. To test methodology adopted and determine compliance with the organization standards by reviewing the documentation produced.
5. To evaluate controls designed for compliance with internal control principles and standards.
6. To determine compliance with common security, auditability and change control standards.

If IS Auditor is part of project team not for performing an audit, but is participating on the project in an advisory role then depending on the level of involvement, IS Auditor may become ineligible to perform audits of the application when it becomes operational.

1.10 SDLC Project Management Techniques and Tools

System Development process may be associated with various automated tools that help in improving productivity and maintaining record and documentation of application being

developed. Tools that help in improving productivity include code generators, development environments (also referred to as developer's workbench) like Visual Studio and Computer-Aided Software Engineering (CASE) applications that help in documenting the SDLC process. In addition, Project Managers may use project management tools like MS Project. This section provides information about these tools. This section covers following three areas:

1. CASE tools
2. Software size estimation covering various techniques used like LOC, FPA analysis etc.
3. Project controlling tools like PERT, CPM and Gantt Charts.

1. Computer-Aided Software Engineering (CASE) tools

SDLC requires collecting, organizing and presenting information required at application systems and program level. This involves building data flows, documenting design of application system, identifying modules/functions/program required to be developed and sometimes developing prototypes to capture requirements. These are essential but time-consuming processes that are required for developing, using and maintaining computer applications.

Computer-Aided Software Engineering (CASE) are automated tools that aid in the software development process. Their use may include tools for capturing and analyzing requirements, software design, code generation, testing, document building and other software development activities.

Although IS Auditor is not expected to have detailed knowledge of how to use CASE tools, they may have to learn how to use CASE tools for effective audit of SDLC project, as required.

Code Generators

Code generators are tools that are a part of CASE tools or development environment like Visual Studio. These tools generate program source code based on parameters provided. These products significantly reduce the development (particularly coding) time; however, maintaining or changing these programs might be painful and time consuming.

Development Environments and Non-Procedural Languages

Developer's Workbench: Provides environment to developer for editing, simulating code, temporary storage, file management and sometimes code generation. It may also provide Software facilities that include the ability to design or paint retrieval screen formats, develop computer-aided training routines or help screens, and produce graphical outputs. It is often referred to as an Integrated Development Environment (IDE).

Non-procedural languages: These are event driven and make extensive use of Object-Oriented Programming concepts such as objects, properties and methods. These languages

cannot perform data intensive or online operations. However, they are best suited to provide an environment to the end user for generating their own views and reports that are required for data analysis and decision making. These languages provide environmental independence (portability) across computer architectures, operating systems and tele-communications monitors. These languages generally have simple language subsets that can be used by less-skilled users.

These languages are classified in the following ways:

1. **Query and Report Generators:** These languages can extract and produce reports and sometimes can access database records, produce complex online outputs.
2. **Embedded Database Languages** are more user-friendly but also may lead to applications that are not integrated well with other production applications.
3. **Relational Database Languages** are usually an optional feature on a vendor's DBMS. These allow the applications developer to make better use of the DBMS product, but they often are not end-user-oriented.

2. Software Size Estimation

Once the work breakdown structure is completed and SDLC methodology (discussed in chapter 4) is finalized Project Manager must perform Software size estimation, i.e. determining the physical size of application (number of programs, modules, reusable function/modules etc.). This helps the Project Manager in deciding resource and skills requirements, to judge the time and cost required for development, and to compare the total effort required by the resources.

Source Lines Of Code (SLOC)

Traditionally, particularly when COBOL like languages was used, software sizing used to be performed using number of Source Lines of Code (SLOC). However, it does not work well for complex systems using different types of programs and automated tools like Source Code Generators. This puts limitation on planning for cost, schedule and quality metrics.

With new technologies, Multi-Point Estimations Techniques were developed that now uses diagrams, objects, spreadsheet cells, database queries and Graphical User Interface (GUI) widgets. These technologies are more closely related to functionality that needs to be created rather than lines of code.

Function Point Analysis (FPA)

The Function Point Analysis (FPA) technique has evolved over the years and is widely used for estimating complexity in developing large business applications. The results of FPA are a measure of the size of an information system based on the number and complexity of the inputs, outputs, files, interfaces and queries with which a user views and interacts with the

data. This is an indirect measure of software size and the process of development. It is based on the number and complexity of inputs, outputs, files, interfaces and queries.

Function points (FPs) are computed by considering various parameters like number of users, number of inputs, number of outputs, expected user actions, data elements to be processed and external interfaces to determine whether a particular module/program is simple, average or complex. This information is used to compute function point using an algorithm that takes into account complexity adjustment values (i.e., rating factors) based on responses to questions related to reliability, criticality, complexity, reusability, changeability and portability.

Function points (FP) derived from this equation are then used as a measure for cost, schedule, productivity and quality metrics (e.g. $\text{Productivity} = \text{FP}/\text{Person-Month}$, $\text{Quality} = \text{Defects}/\text{FP}$, and $\text{Cost} = \text{Monetary Value}/\text{FP}$).

IS Auditor should be familiar with the use of Function Point Analysis. However, IS Auditors are not expected to be experts in this technique.

FPA Feature Points

In web-enabled applications, the development effort depends on the number of forms, number of images; type of images (static or animated), features to be enabled, interfaces and cross-referencing that is required. Thus, from the point of view of web applications, the effort would include all that is mentioned under Function Point Estimation, plus the features that need to be enabled for different types of user groups. The measurement would involve identification or listing of features, access rules, links, storage, etc.

A slightly different approach for System Software such as Operating Systems, Telephone Switching Systems, etc. was developed. To differentiate from FPA it is called "Feature Points". It is used for software that has well-defined algorithms like Systems Software, Embedded Software, Real Time Software, CAD, Artificial Intelligence and some traditional MIS software.

Cost Budgets

Cost estimates of a SDLC project are based on the amount of effort likely to be required to carry out each task. The estimates for each task contain one or more of the following elements:

1. Person-hours for all type of resources e.g. System Analyst, Programmers, Support Staff, Testing Teams etc. (Pl. refer section 3.9 roles and responsibilities)
2. Infrastructure (Hardware, Software, Networks etc.), other specialized software, if any and communication equipment
3. Other costs such as third-party services, automation tools required for the project, consultant or contractor fees, training costs, etc.

Based on estimates following steps are used in arriving at cost budget:

- Prepare estimate of human and machine effort by for all tasks.
- Determine hourly rate for each type of person-hours and arrive total person cost.

3. Project Controlling Tools and Techniques

Project Manager uses various tools and techniques to control the project. The graphical techniques used to represent schedule are:

1. Project Evaluation Review Technique (PERT)
2. Critical Path Method (CPM)
3. Gantt Chart

A. Program Evaluation Review Technique (PERT)

PERT is a technique to estimate the efforts and time required to complete the work/task described by Work Breakdown Structure (WBS). Project Manager lists out the major tasks and arrives at three different duration estimates of each activity. The three estimates are then used to derive single estimate applying a mathematical formula. PERT is often used in projects with uncertainty about the duration. Table 1.1 illustrates one such formula for a hypothetical project where activities are named from A to L. The first is the most optimistic time (if everything went well) and the third is the pessimistic or worst-case scenario. The second is the most likely scenario. This estimate is based on experience attained from projects that are similar in size and scope. To calculate the PERT time, estimate for each given activity, the following calculation is applied: $[\text{Optimistic} + \text{Pessimistic} + 4(\text{most likely})]/6$

Table 1.1: PERT table

Activity	Optimistic Estimate	Pessimistic Estimate	Most Likely Estimate	Final Estimate
A	2	6	4	4
B	4	10	7	7
C	4	14	9	9
D	7	19	10	11
E	2	6	4	4
F	1	5	3	3
G	2	8	5	5
H	3	9	6	6
I	2	6	4	4
J	1	5	3	3
K	2	6	4	4
L	1	3	2	2

Figure 1.4 illustrates use of the PERT Network Management Technique. (Each circle represents milestones and the arrow represents activities. Number after activity shows the number of days required to complete the activity.)

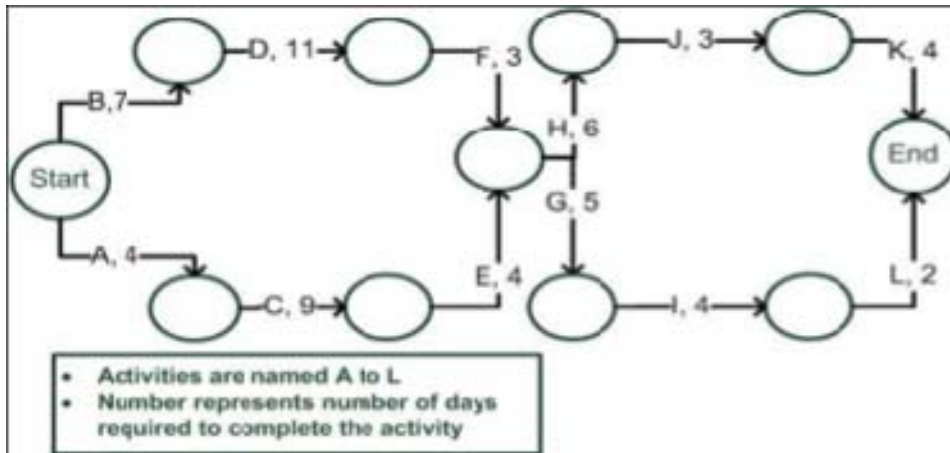


Figure 1.4: PERT Diagram

Some Project Managers show all estimates in the PERT diagram.

B. Critical Path Methodology

All project schedules have a critical path. Activities of a project are in sequence or independent or parallel. A project can be represented as a network of activities is shown in PERT diagram. (Some Project Managers refer to PERT diagram as PERT Network).

A path through the network is any set of successive activities which go from the beginning to the end of the project. Associated with each activity in the network is a single number that represents estimates the amount of time that the activity will require to complete.

The critical path is the sequence of activities whose sum of activity time is highest than that for any other path through the network. Critical paths represent the shortest possible project completion time, if everything goes according to schedule. In other words, delay in completing any activity on critical path delays the overall project.

Activities which are not in the critical path have slack time, i.e. delay in performing these activities may not affect the overall project schedule. Activities on critical path have zero slack time.

The PERT diagram shown in figure 1.4 has following 4 paths:

1. A – C – E – G – I – L
2. A – C – E – H – J – K

3. B – D – F – H – J – K

4. B – D – F – G – I – L

Using the time estimates the total time required for each path is 28, 30, 34 and 32 days respectively. Third path hence is Critical Path. (Shown by thick arrows in figure 1.5)

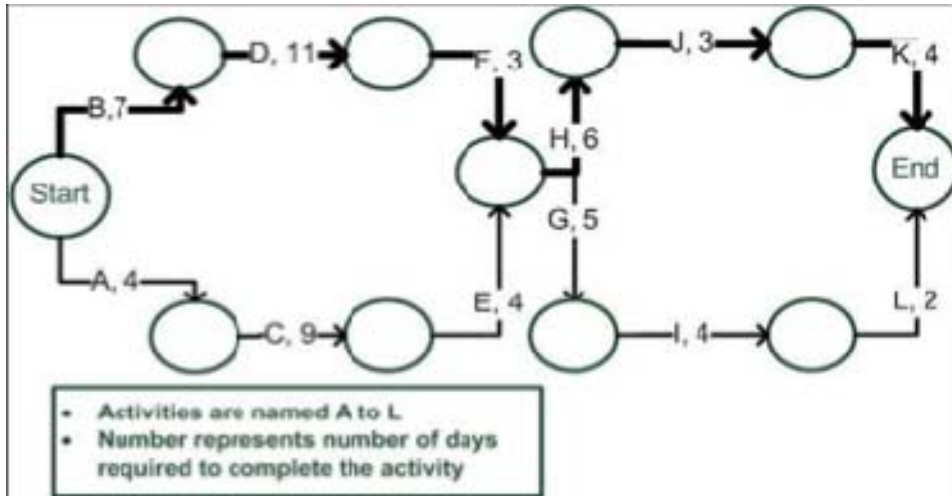


Figure 1.5: Critical Path Method (CPM)

Project Manager can use the slack time on non-critical path for scheduling resources optimally, since slack time provides flexibility to start activity late than scheduled start date. The slack times for a project are computed by working forward through path, computing the earliest possible completion time for each activity, until the earliest possible completion time for the total project is found. Then by working backward through the network, the latest completion time for each activity is found, the slack time computed and the critical path identified.

Most CPM packages facilitate the analysis of resource utilization per time unit (e.g., day, week, etc.) and resource levelling, which is a way to level off resource peaks and valleys.

C. Gantt Charts

Gantt Charts are aid for scheduling activities/tasks needed to complete a project. These charts show details related to activities calculated during PERT and CPM. The charts also show which activities are in progress concurrently and which activities must be completed sequentially. Gantt Charts may reflect the resources assigned to each task and by what percent allocation. The charts aid in identifying activities that have been completed early or late. Progress of the entire project can be tracked from the Gantt Chart. Gantt Charts can also be used to track the milestones for the project.

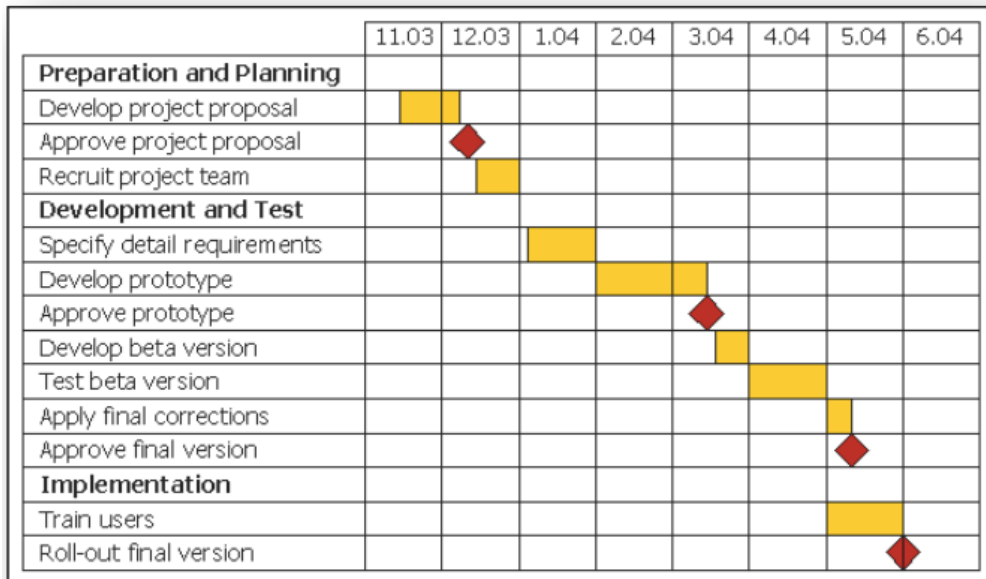


Figure 1.6: Gantt chart

1.11 Summary

Every project has unique success criteria based on the expectations of stakeholders. Generally, success criteria are measurable and manageable such as cost, time and scope. However, some criteria, such as meeting business needs, are subjective but essential. The project sponsor is a key stakeholder who defines such success criteria. The project team should capture project requirements and document them at the initial stage to complete the project successfully. Activity of capturing requirements is usually difficult because it involves subjective decisions and extensive interaction between users and developers. Requirements should be formally approved and then frozen (baselined) to prevent Scope Creep. Success criteria allow the Project Manager to focus on managing risks that can affect desirable outcome and successful completion of the project.

IS Auditor should review adequacy of the following project management activities:

- Levels of oversight by Project Committee/Board
- Risk Management methods within the project
- Issue Management
- Cost Management
- Processes for Planning and Dependency Management

- Reporting processes to Senior Management
- Change Control processes
- Stakeholder Management involvement
- Sign-off process
- Adequate documentation of all phases of the SDLC process such as:
 - Availability of clearly defined objectives on what is to be accomplished during each phase.
 - Key deliverables of each phase with project personnel assigned direct responsibilities for these deliverables.
 - A project schedule with highlighted dates for the completion of key deliverables.
 - An economic forecast for each phase, defining resources and the cost of the resources required to complete the phase.

1.12 Questions

1. Who among the following is responsible for ongoing facilitation of a SDLC project?
 - A. Project Sponsor
 - B. Project Manager
 - C. Steering Committee
 - D. Board of Directors
2. A Multi-National organization has decided to implement an ERP solution across all geographical locations. The organization shall initiate a:
 - A. Project
 - B. Program
 - C. Portfolio
 - D. Feasibility study
3. Which of the following primarily helps Project Manager in mitigating the risk associated with change in scope of software development project?
 - A. Change Management Process
 - B. Use of Prototyping

- C. Revising Effort Estimates
 - D. Baselining requirements
4. Monitoring which of the following aspect of SDLC project shall help organization in benefit realization over sustained period of time?
- A. Quality
 - B. Budget
 - C. Schedule
 - D. Methodology
5. Which of the following tools and techniques primarily help in improving productivity of SDLC project team members?
- A. Use of Standard Methodology
 - B. Software Sizing using FPA
 - C. Developers' Workbench
 - D. Appropriate HR Policies
6. While performing mid-term review of SDLC project, the IS Auditor primarily focuses on:
- A. Project Risk Management Process
 - B. Adherence to the schedule
 - C. Reviewing minutes of Steering Committee Meeting
 - D. Cost Management is as per budget
7. A Project Manager's main responsibility in a project meant to create a product is:
- A. Ensuring it is high grade
 - B. To pack exciting features in the product
 - C. Ensuring it is high quality
 - D. Creating a product within allocated cost and schedule
8. The Project Manager should be able to fulfill the role of:
- A. An Integrator
 - B. A Functional Manager

- C. A Line Manager
 - D. A Sponsor
9. The most successful Project Manager usually:
- A. Works his/her way up from Assistants in the project office to full-fledged Project Managers, supplementing that experience with formal education.
 - B. Comes right from Harvard's MBA program into managing very large projects.
 - C. Are the Technical Experts.
 - D. Have considerable experience as a Functional Manager before moving into the Project Management arena.

1.13 Answers and Explanations

1. A is the correct answer. Project Sponsor is a stake holder having maximum interest / stake in the success of project and his primary responsibility is to coordinate with various stakeholders for success of project. Option B: Project Manager is responsible for executing the project activities. Option C: Steering Committee monitors project progress but is not ongoing activity. Option D: Board of Directors provides direction.
2. B is the correct answer. A program is concerned with the benefits received, from implementing it, whereas project deals with specific deliverables. The scope of the program is wider in comparison to the project. The project works on a single functional unit, while the program works on various functional units. A portfolio contains both projects and programs and is managed by a portfolio manager. Option D: Feasibility study either has been completed or shall be initiated as part of program.
3. D is the correct answer. Scope Creep of continued changes in requirements during SDLC project is most common risk. If not properly handled the project may be delayed and benefit realization from the project shall be affected. The Project Manager therefore, must freeze the scope by base-lining requirements. Any change after base-lining shall follow. Option A: Change Management process without base-lining may not help. Project Manager may or may not. Option B: is used for freezing the requirements. Option D: revised effort estimate is applicable after change is approved.
4. A is the correct answer. Quality is most important aspect for SDLC project, since it minimizes errors that can impact operations. Options B, C and D are of prior to monitoring phase.
5. C is the correct answer. Automated tools help team in improving productivity as these tools help in managing mundane and structure activities and developers can focus on core activities. Developers' workbench provides various functions that help in improving

productivity. Option A: Use of standards help in following uniform methods and reducing rework. Option B: Software Sizing is the main input parameter to cost estimation models. Option D: HR policies may help in motivating team but it is secondary.

6. A is the correct answer. Auditor should primarily focus on risk management that will provide inputs on events that has impact on all aspects of project. Options B, C and D help in confirming the findings from review of Risk Management process.
7. C is the correct answer. A Project Manager is responsible to ensure high quality in a way that the final product meets the specifications and quality benchmarks. Options A, B and C are not the main responsibility of a Project Manager.
8. A is the correct answer. The Project Manager is responsible for collective project success. The Project Manager integrates a project as a whole. He/she unifies various aspects and processes of initiating, planning, executing, monitoring, control and closure. Options B, C and D is not the role of the Project Manager.
9. A is the correct answer. A Project Manager must have experience in working on projects in various roles including the role of a Project Manager. Options B, C and D are secondary aspect.

Chapter 2

SDLC – Need, Benefits and Phases

Learning Objectives:

After completion of this chapter you should have conceptual clarity on the basic concepts of System Development Life Cycle (SDLC), changes in SDLC due to change of technology and business environment. This chapter will also help to understand the inclusion of newer phases in SDLC. This chapter covers:

- Traditional SDLC phases and overview of the main activities;
- Additional phases due to availability of outsourcing and generic customizable software; and
- Steps added in different phases due to security requirements (Secure SDLC or SSDLC).

2.1 What is SDLC?

SDLC refers to the process of examining a business case with the intent of improving it through better procedures and methods. This is required when there is need to change business processes due to requirements arising out of customers/stakeholder's expectations and business strategy. These changes are generally attributed to need to automate the service delivery using information and related technology. System development involves developing or acquiring and maintaining application systems that are used for various day-to-day business process activities. Generally, these systems process data of business transactions.

A standard set of steps used for developing systems is called a SDLC. SDLC generally uses various methods depending upon the type and nature of application. For example, a batch processing application that processes historical data to generate reports for management's information may use a model where activities are performed one after another (waterfall model), where as if there is no clear understanding of what functions can be automated, an iterative (spiral) model may be used. Similarly, depending upon the availability of skilled resources, the development team may adopt different methodology for developing software.

2.2 Relevance of SDLC for Business Process Automation

Business Application System, also called Application Software, is designed to support a specific function or process of an organization, such as management of inventory, payroll, or analysis of market. The objective of application system is to process data to produce

information. For example, a software developed for the managing inventory at a bookstore may keep track of the inventory of books in stock for the latest bestseller. Application System for the Human Resource Department may keep track of the changing payroll information of the employees.

System Development involves developing or acquiring and maintaining Application Systems which are used for various day-to-day business activities. These business activities are called as Business Processes and they process data. The effective management and control of this System Development is critical as the business systems process and control information assets of the organization. The use of standard set of steps to develop and support business applications is called Systems Development Methodology.

2.3 Need for SDLC

The need for business development or acquisition of new applications may arise to due to following situations:

- New service delivery opportunity that relates to a new or existing business process (e.g. e-commerce);
- Issues and problems with an existing systems/business process (complaints from customers/users);
- Change in strategic focus leading to an opportunity that will provide benefits to the organization (Mergers and Acquisitions, or new Service Delivery Channels like ATM for Banks);
- New opportunity due to advancement of existing technology or availability of new technology (e.g. use of Mobile Technology for Banking Services); and
- Use of automation by competitors to enhance quality of services.

All of these situations directly affect the business drivers. Business drivers can be stated as the attributes of a business function (service delivery) that arise out of strategic objectives to enhance targets and goals of business function to achieve the strategic goals of the business. In other words, business objectives defined by strategy gets translated into drivers for business operations which require new application software or upgrading of existing application software. This results in initiating an SDLC project.

2.4 Benefits of SDLC

There are many benefits for deploying a SDLC including the ability to pre-plan and adopt a structured approach for its' phases and goals. The goal-oriented processes of SDLC are comprehensive in applicability and can be modified to meet changing needs. However, if SDLC is well-defined for the business, one can:

- Have a clear view of the entire project, the personnel involved, staffing requirements, a defined timeline, and precise objectives to close each phase.
- Base costs and staffing decisions on concrete information and need.
- Provide verification, goals, and deliverables that meet design and development standards for each step of the project, developing extensive documentation throughout.
- It usually begins with the analysis of cost and timelines and provides developers a measure of control through the phased and iterative approach.
- Improvement may be brought in the quality of the final system through verification at each stage.

2.5 Phases of SDLC

System Development life cycle is a sequence of activities performed by group of users and IT Development experts. These set of activities are grouped together to form phases and generally each phase has pre-determined set of deliverables and/or a milestone to be reached. Typically, a SDLC consists of 7 phases.

The number of phases might vary for each SDLC project depending upon milestones and/or deliverables. For example: if an organization is developing a software using internal development team, it may not lay much emphasis on User Acceptance Testing (UAT) and may club this activity with Testing Phase, whereas in case of outsourced development or acquired software, UAT is a major milestone and has pre-defined deliverables that are signed off. In the diagram below, considering the criticality of outsourced applications, UAT has been shown as a separate phase

2.5.1 Phase 1: Feasibility Study

The feasibility study is based on technical, economical and social aspects and this helps in determining strategic benefits of using system. These benefits can be either in productivity gains or in future cost avoidance. The study has to also identify and quantify the cost savings and estimate the probable Return on Investment. This information is used to build a business case covering both tangible as well as intangible factors such as readiness of the business users and maturity of the business processes. The business case provides inputs for business justification for moving to the next phase and is also used for reviewing progress or evaluating success of SDLC project.

Detailed steps of feasibility study are discussed in chapter 2. The feasibility study shall be different for different applications depending upon the expected benefits for the organization. For example, if an organization intends to implement an application that is already implemented by various organizations of similar type, it may use a generic feasibility study

and focus only on the benefits to the organization, such as providing Internet Banking services or Mobile Banking services.

Role of IS Auditor in project initiation and feasibility study phase:

- Review of documentation for the reasonableness.
- Review cost justification/benefits with schedule of when the anticipated benefits may be realized.
- Identify if the business needs used to justify the system actually exist.
- Justification for going for a development or acquisition.
- Review the alternate solutions for reasonableness.
- Review the reasonableness of the chosen solution.

2.5.2 Phase 2: Requirements Definition

This phase involves preparing the statement of intent explaining the problem or the need for new application to provide functional, service and quality requirements of the solution system. The user needs to be actively involved in requirements definition. This involves:

- Studying needs of the users
- Obtaining inputs from employees and managers on their expectations
- Determining information requirements of the users

Several fact-finding techniques and tools such as questionnaires, interviews, observing decision-maker behaviour and their office environment etc. are used for understanding the requirements.

Role of IS Auditor in Requirements Definition phase:

- Identify the affected users and the key team members on the project to verify that they are having an appropriate representation.
- Review detailed requirements definition document and verify its accuracy and completeness through interviews with the affected and requested user departments.
- Review existing Data Flow Diagrams (DFD) and other related specifications like forms, Data Descriptions, Output Formats, etc., to ensure that they cover the user requirements.

2.5.3 Phase 3a: System Analysis

This refers to the process of gathering and analyzing the facts, diagnosing problems, and using the outcome to recommend improvements to the proposed system. Before arriving at

new design, one must thoroughly understand existing process/system and map them against new requirements to understand changes and rationale for changes. Analysis is also important to decide upon system design approach. Traditional system development generally adopts a data oriented approach, since it had been focused on processing and presenting of business data., However, due to extensive use of technology in modern organizations, the focus now is more on service oriented approach where the objective of the system is to provide services using data models.

Role of IS Auditor in System Analysis phase:

- Verify that Management has approved the initiation of the project and the cost.
- In case of acquisition, determine that an appropriate number of vendors have been given proposals to cover the true scope of the project and requirements of the users.
- Determine whether the application is appropriate for the user of an embedded audit routine or modules and if so, request may be made to incorporate the routine in conceptual design of the system.

2.5.4 Phase 3b: Design

This phase takes primary inputs from *Phase 1, i.e. Requirement Definition*. Based on the requirements identified, the team may need to finalize requirements by multiple user interactions and establish a specification baseline for development of system and subsystem.

These specifications describe:

- Parts of the System
- How they interface
- How the System need to be implemented
- Type of Hardware, Operating System and other Software
- Network facilities
- Program and Database Specifications
- Security considerations

Additionally, a formal change management process should be established to prevent uncontrolled entry of new requirements during development process.

Role of IS Auditor in design phase:

- Review system flowcharts for adherence to the general design
- Review input, processing and output controls and ensure that they have been appropriately included in the system.

- Assess adequacy of the audit trails which provide traceability and accountability.
- Verify key calculations and processes for correctness and completeness.
- Interview users to ascertain their level understanding of the system design, input to the system, screen formats and output reports.
- Verify that system can identify erroneous data correctly and can handle invalid transactions.
- Review conceptual design to ensure the existence of appropriate controls.
- Review quality assurance and quality control results of programs.
- Verify the design for its completeness and correctness and ensure that it meets the defined requirements.
- Verify that the functional data created during requirement phase is complete and test plans are developed.

2.5.5 Phase 4: Development

In this phase, efforts are made to use the design specifications to begin programming, and formalizing support for operational processes of the system. After the system design details are resolved, the resource needs such as specific type of hardware, software, and other services are determined. The choices depend on many factors such as time, cost and availability of skilled resources, i.e. programmers and testers. The analyst works closely with the programmers. During this phase, the analyst also works with users to develop required documentation for software, including various procedure manuals. In the development phase, the design specifications are converted into a functional system that will work in planned system environment. Application programs are written, tested and documented. Finally, this results in development of a fully functional and documented system. A very well coded application program should have the following characteristics:

- **Reliability:** It refers to the consistency with which a program operates over a period of time. However, poor setting of parameters and hard coding of some data subsequently could result in the failure of a program after some time.
- **Robustness:** It refers to the applications' strength to perform operations in adverse situations by taking into account all possible inputs and outputs of a program considering even the least likely situations.
- **Accuracy:** It refers not only to what program is supposed to do', but also the ability to take care of 'what it should not do'. The second part is of great interest for quality control personnel and auditors.

- **Efficiency:** It refers to the performance per unit cost with respect to relevant parameters and it should not be unduly affected with the increase in input values.
- **Usability:** It refers to a user-friendly interface and easy-to-understand internal/external documentation.
- **Readability:** It refers to the ease of maintenance of program even in the absence of the program developer.

Some key aspects of development:

1. **Program Coding Standards:** The logic of the program outlined in the flowcharts is converted into program statements or instructions. For each language, there are specific rules concerning format and syntax. Syntax means vocabulary, punctuation and grammatical rules available in the language manuals that the programmer has to follow strictly and pedantically. Different programmers may write a program using different sets of instructions but each giving the same results. This might create a problem for changes to be done to the program which has been written by another programmer. Therefore, the coding standards are to be defined so as to serve as a method of communication between teams, amongst the team members and users resulting in better controls. Coding standards minimize the system development issues due to programmer turnover. These standards provide simplicity, interoperability, compatibility, efficient utilization of resources and reduce processing time.

2. **Programming Language:** Depending upon the development approach, the analyst decides the programming language to be used. Application programs are coded in the form of statements or instructions and the same is converted by the compiler to object code for the computer to understand and execute. The programming languages commonly used are:

- High-level general-purpose programming languages such as COBOL and C;
- Object oriented languages such as C++, JAVA etc.;
- Scripting language such as JavaScript, VBScript; and
- Decision Support or Logic Programming languages such as LISP and PROLOG.

The choice of a programming language may depend on various pertinent parameters. In general, language selection may be made on the basis of application area; algorithmic complexity; environment in which software has to be executed; performance consideration; data structure complexity; knowledge of System Development staff; and capability of in-house staff for maintenance.

Role of IS Auditor in development phase:

- Ensure that documentation is complete.
- Review QA report on adopting coding standards by developers.
- Review the testing and bugs found are reported and sent for rework to developers.

2.5.5.1 Software Escrow

A Software Escrow arrangement requires the developer of a software product to place proprietary materials necessary to maintain the product in escrow with a neutral party known as the Escrow Agent. Should the software vendor, or licensor, fail to support the product, the Escrow Agent agrees to release the proprietary materials (such as source code) to the end-user. The end-user or licensee is then allowed to employ the deposit materials to support the licensed product. A partial list of recommended deposit materials includes:

- Two copies of the Source Code for each version of the licensed software on magnetic media
- All manuals not provided to the licensee (technical, operator/user, installation)
- Maintenance tools and necessary third-party system utilities
- Detailed descriptions of necessary non- licensor proprietary software, descriptions of the programs required for use and/or support that the developer does not have the right to offer to the licensee
- Names and addresses of key technical employees that a licensee may hire as a sub-contractor in the event the developer ceases to exist
- File listings generated from any magnetic media
- Compilation instructions in written format or recorded on video format.

2.5.6 Phase 5: Testing

Before the information system can be used, it must be tested. Systems testing are done at various stages during development till implementation. There are primarily two types of testing:

1. Quality Assurance Testing, that includes Unit Testing, Interface Testing, Integration Testing and Peer Reviews.
2. User Acceptance Testing (UAT) also known as Final Acceptance Testing.

Testing establishes the actual operation of the new information system, with the final iteration of User Acceptance Testing and user sign-off. Organization may consider going for a certification and accreditation process to assess the effectiveness of the business application. This provides assurance to the management about:

- Mitigating risks to an appropriate level.
- Providing accountability over the effectiveness of the system in meeting objectives.
- Establishing an appropriate level of internal control.

UAT supports the process of ensuring that the system is production-ready and satisfies all documented requirements. The methods include:

- Definition of test strategies and procedures.
- Design of test cases and scenarios.
- Execution of the tests.
- Utilization of the results to verify system readiness.

Acceptance criteria are defined so that a deliverable satisfies the pre-defined needs of the user. A UAT plan must be documented for the final test of the completed system. The tests are written from a user perspective and should test the system in a manner as close to production as possible. For example, tests may be based around typical pre-defined, business process scenarios. If new business processes have been developed to accommodate the new or modified system they should also be tested at this point. A key aspect of testing should also include testers seeking to verify that supporting processes integrate into the application in an acceptable manner. Successful completion would generally enable a project team to hand over a complete integrated package of application and supporting procedures.

Ideally, UAT should be performed in a secure testing or staging environment. A secure testing environment where both source and executable code are protected helps to ensure that unauthorized or last-minute changes are not made to the system without going through the standard system maintenance process. The nature and extent of the tests will depend on the magnitude and complexity of the system change.

Testing primarily focuses on ensuring that the software does not fail i.e. it will run according to its specifications and in the way users expect. Special test data are input for processing and the results are examined against pre-determined output. If it is found satisfactory, it is eventually tested with actual data from the current system.

Role of IS Auditor in testing phase:

- Review the test plan for completeness and correctness.
- Review whether relevant users have participated during testing phase.
- Review error reports for their precision in recognizing erroneous data and for resolution of errors.
- Verify cyclical processes for correctness (example: year-end process, quarter-end process)
- Interview end-users of the system for their understanding of new methods, procedures and operating instructions.

- Review the system and end-user documentation to determine its completeness and correctness.
- Review whether reconciliation of control totals and converted data has been performed to verify the integrity of the data after conversion.
- Review all parallel testing results.
- Test the system randomly for correctness.
- Review unit test plans and system test plans to determine that tests for internal control are addressed.
- Verify that the system security is functioning as designed by developing and executing access tests.
- Ensure test plans and test results are maintained for reference and audit

2.5.7 Phase 6: Implementation

This involves roll out of the application which has been developed or acquired for the business function based on the current state. The approach for implementation will be decided based on this state. One of the following approaches may be adopted: (This is *discussed in more detail in chapter 6*)

1. **Cut-off:** Where old system/process is discontinued and new application is made live (operational).
2. **Phased implementation:** Where new application is started in logical phases for different functions.
3. **Pilot:** Where a part function is implemented using new application and based on result either phased or cut-off approach is followed.
4. **Parallel:** Where both the old and new system run simultaneously and based on problem resolution and reliability of processing by the new system, the old system is discontinued.

Role of IS Auditor in Implementation Phase:

- Ensure that test plans, test data and test results are maintained for reference and audit.
- Determine that the formal acceptance has been signed by the Project Development team, User Management team, Quality Assurance team and Security Professional/Auditor.
- Verify that the system has been installed according to the organization's change control procedures.

- Review programmed procedure used for scheduling and running the system along with the system parameters that are used in executing the production schedule.
- Review all system documentation to ensure its completeness and verify whether all recent updates from the testing phase have been incorporated.
- Verify that data conversion is correct and complete and is confirmed by the respective User Departments before the system is implemented and Final User Sign-off is obtained.

2.5.8 Phase 7: Maintenance

This is the post-implementation stage following the successful implementation of a new or extensively modified system. This requires, implementation of a formal process that:

1. Provides support and assistance to users in smooth operations and end-user management.
2. There is a mechanism to record, review and implement deficiencies and future changes required.
3. Assess adequacy of the system and projected ROI measurements as per business case.
4. Update project management process based on lessons learned and recommendations for future projects regarding system development.

Role of IS Auditor in Maintenance and Post-Implementation Phase:

Sufficient time should be allowed before post-implementation review for the system to stabilize in the live environment. Then only there may be significant problems that would have surfaced. Some prominent roles of the IS Auditor are:

- Determine that the systems objective requirements were achieved
- Determine if the cost benefits identified in the feasibility study are being measured.
- Review that the required controls have been built into the system to ensure that they are operating as designed.
- Review error logs to determine if there is any resource or operating problems inherent with the system. Logs may indicate the inappropriate planning or testing of the system prior to implementation.
- Review input and output control balances and reports to verify that system is processing data correctly and completely.
- Evaluate adequacy of procedures for authorizing, prioritizing and tracking system changes.

- Identify system changes and verify that appropriate authorization was given to make the change in accordance with organizational standards.
- Review permanent program documentation to ensure that evidence (Audit Trail) is retained regarding program changes.
- Evaluate adequacy of the security access restrictions over production source and executable modules.
- Evaluate adequacy of the organization's procedures for dealing "emergency" program changes.
- Evaluate the adequacy of the security access restrictions over the use of the "emergency" logon-ids.
- Verify existence and adequacy of the records for system changes.
- Evaluate adequacy of the access protection of maintenance records.

2.6 Types of SDLC Model

2.6.1 Waterfall Model

The Waterfall Approach is a traditional development approach in which each phase is executed in sequence or in linear fashion. These phases include requirements analysis, specifications and design requirements, coding, final testing, and release. Fig. 2.1 shows representative model of this method. When the traditional approach is applied, an activity is undertaken only when the prior step is completed.

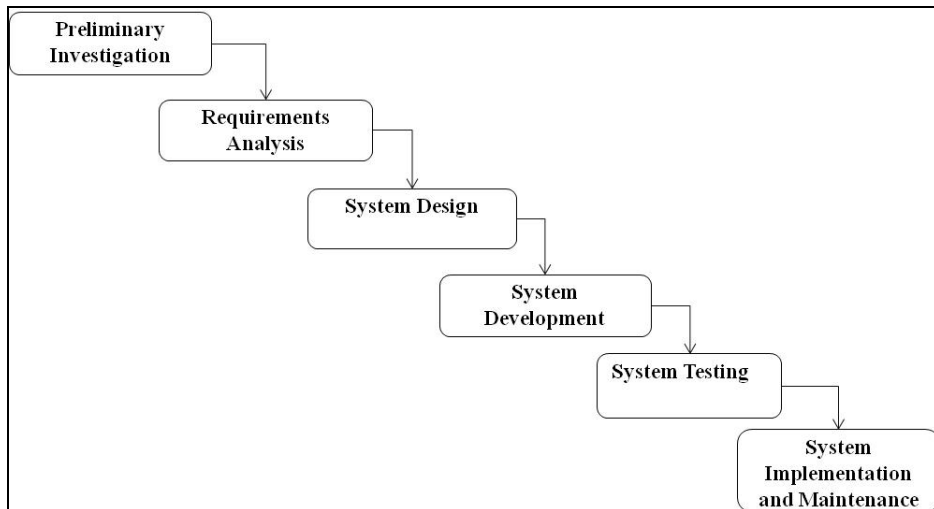


Fig. 2.1: Waterfall Approach

The characterizing features of this model have influenced the development community in big way. Some of the key characteristics are:

- Project is divided into sequential phases, with some overlap and splash back acceptable between phases.
- Emphasis is on planning, time schedules, target dates, budgets and implementation of an entire system at one time.
- Tight control is maintained over the life of the project through the use of extensive written documentation, as well as through formal reviews and approval/signoff by the user and information technology management occurring at the end of most phases before beginning the next phase.

Strengths:

- It is ideal for supporting less experienced project teams and Project Managers, or project teams whose composition fluctuates.
- The orderly sequence of development steps and design reviews help to ensure the Quality, Reliability, Adequacy and Maintainability of the developed software.
- Progress of system development can be tracked and monitored easily.
- It enables to conserve resources.

Weaknesses:

- It is criticized to be Inflexible, slow, costly, and cumbersome due to significant structure and tight controls.
- Project progresses forward, with only slight movement backward.
- There is a little to iterate, which may be essential in situations.
- It depends upon early identification and specification of requirements, even if the users may not be able to clearly define 'what they need early in the project'.
- Requirement inconsistencies, missing system components and unexpected development needs discovered during design and coding are most difficult to handle.
- Problems are often not discovered until system testing.
- System performance cannot be tested until the system is almost fully coded, and under capacity may be difficult to correct.
- It is difficult to respond to changes, which may occur later in the life cycle, and if undertaken it proves costly and are thus discouraged.
- Written specifications are often difficult for users to read and thoroughly appreciate.
- It promotes the gap between users and developers with clear vision of responsibility.

2.6.2 Incremental Model

The Incremental model is a method of software development where the model is designed, implemented and tested incrementally (a little more is added each time) until the product is finished. The product is defined as finished when it satisfies all of its requirements. This model combines the elements of the waterfall model with the iterative philosophy of prototyping. It is pictorially depicted in Fig. 2.2.

The product is decomposed into a number of components, each of which are designed and built separately (termed as Builds). Each component is delivered to the client when it is complete. This allows partial utilization of product and avoids a long development time. It also creates a large initial capital outlay, and the subsequent long wait is avoided. This model of development also helps to ease the traumatic effect of introducing completely new system all at once. A few pertinent features are listed as follows:

- A series of mini-Waterfalls are performed, where all phases of the Waterfall development model are completed for a small part of the system, before proceeding to the next increment.
- Overall requirements are defined before proceeding to evolutionary, mini-Waterfall development of individual increments of the system.
- The initial software concept, requirement analysis, and design of architecture and system core are defined using the Waterfall approach, followed by Iterative Prototyping, which culminates in installation of the final prototype (i.e. working system).

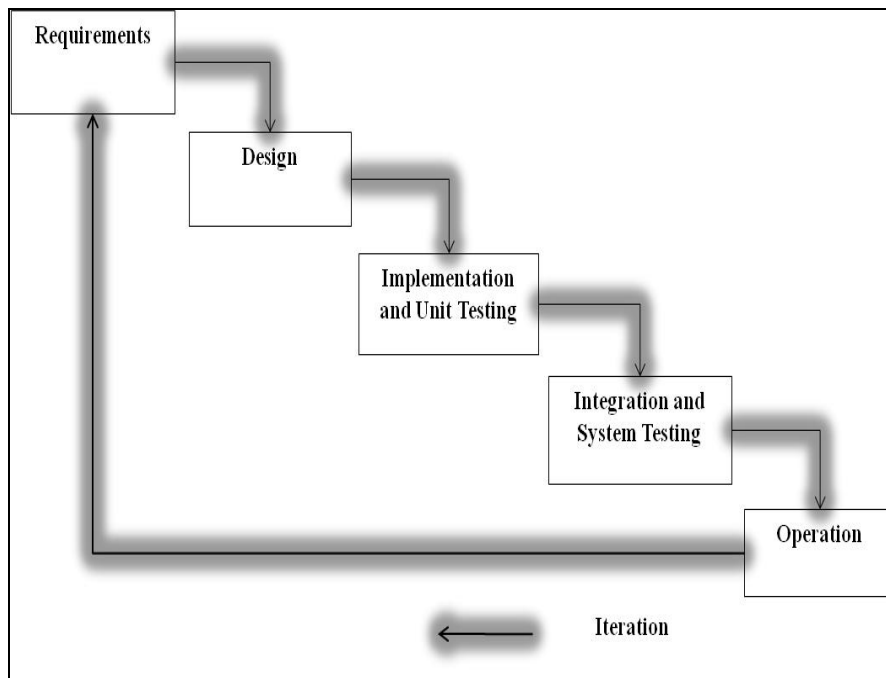


Fig. 2.2: Incremental Model

Strengths:

- Potential exists for exploiting knowledge gained in an early increment as later increments are developed.
- Moderate control is maintained over the life of the project through the use of written documentation and the formal review and approval/signoff by the user and information technology management at designated major milestones.
- Stakeholders can be given concrete evidence of project status throughout the life cycle.
- It is more flexible and less costly to change scope and requirements.
- It helps to mitigate integration and architectural risks earlier in the project.
- It allows the delivery of a series of implementations that are gradually more complete and can go into production more quickly as incremental releases.
- Gradual implementation provides the ability to monitor the effect of incremental changes, isolated issues and make adjustments before the organization is negatively impacted.

Weaknesses:

- When utilizing a series of mini-Waterfalls for a small part of the system before moving onto the next increment, there is usually a lack of overall consideration of the business problem and technical requirements for the overall system.
- Each phase of an iteration is rigid and do not overlap each other.
- Problems may arise pertaining to system architecture because not all requirements are gathered up front for the entire software life cycle.
- Since some modules will be completed much earlier than others, well-defined interfaces are required.
- It is difficult to demonstrate early success to management.

2.6.3 Software Reengineering and Reverse Engineering

Software Reengineering

Reengineering as name suggest is a process of updating an existing system by reusing design and program components. Although it updates existing software as it is used in case of major changes in existing system, it differs from change management due to the extent of changes. These changes typically prompt for new software development project, however as an interim solution a reengineering project is initiated. A number of tools are now available to support this process.

What is SoftwareReengineering?

- Restructuring or rewriting part or all of a system without changing its functionality
- Applicable when some (but not all) subsystems of a larger system require frequent maintenance
- Reengineering involves putting in the effort to make it easier to maintain
- The reengineered system may also be restructured and re-documented

When to Reengineer?

- When system changes are confined to one subsystem, the subsystem needs to be reengineered
- When hardware or software support becomes obsolete
- When tools to support restructuring are readily available
- When some business processes or functions are reengineered

Software Reengineering Activities

- **Inventory Analysis** – Listing and identifying active software applications and components required by business. The attributes of applications can be criticality, longevity, current maintainability. This helps in identifying reengineering criteria.
- **Document Restructuring** – Identify documentation for identified applications/modules. (Identifying poor or weak documentation for re-documentation sometimes considered as reengineering activity)
- **Design Recovery** – Identify the design of the application module to be reengineered. (In case it is not available it may have to be built based on code or using Reverse Engineering method)
- **Reverse Engineering** – Process of design recovery - Analyzing a program in an effort to create a representation of the program at some abstraction level higher than source code
- **Code Restructuring** – Source code is analysed and violations of structured programming practices are noted and repaired, the revised code also needs to be reviewed and tested
- **Data Restructuring** – Usually requires full Reverse Engineering, current data architecture is dissected and data models are defined, existing data structures are reviewed for quality
- **Forward Engineering** – also called reclamation or renovation, recovers design information from existing source code and uses this information to reconstitute the existing system to improve its overall quality and/or performance.

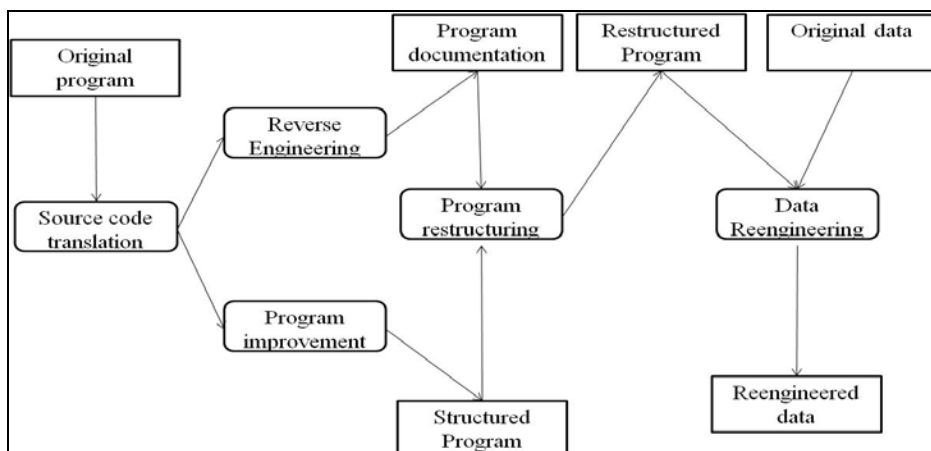


Figure 2.3: Reengineering and reverse Engineering

Reverse Engineering

Reverse Engineering is the process of studying and analysing an application, a software application or a product to see how it functions and to use that information to develop a similar system. This process can be carried out in several ways:

- Decompiling object or executable code into source code and using it to analyze the program
- Black Box testing: The application to be reverse-engineered to unveil its functionality

The major advantages of Reverse Engineering are:

- Faster development and reduced SDLC duration
- The possibility of introducing improvements by overcoming the reverse-engineered application drawbacks

The IS Auditor should be aware of following risks:

- Software license agreements often contain clauses prohibiting the licensee from reverse engineering the software so that any trade secrets or programming techniques are not compromised.
- De-compilers are relatively new tools with functions that depend on specific computers, operating systems and programming languages. Any change in one of these components may require developing or purchasing a new de-compiler.

2.6.4 Object Oriented Software Development (OOSD)

OOSD differs from traditional SDLC approach which considers data separately from the procedures that act on them (e.g., program and database specifications). OOSD is the process of solution specification and modelling where data and procedures can be grouped into an entity known as an object. An object's data are referred to as its attributes and its functionality is referred to as its methods. Proponents of OOSD claim the combination of data and functionality is aligned with how humans conceptualize everyday objects.

Objects usually are created from a general template called a Class. The template contains the characteristics of the Class without containing the specific data that need to be inserted into the template to form the Object.

Classes are the basis for most design work in Objects. Classes are either Super-Classes (i.e., Root or Parent Classes) with a set of basic attributes or methods, or Sub-Classes which inherit the characteristics of the Parent Class and may add (or remove) functionality as required. In addition to inheritance, Classes may interact through sharing data, referred to as aggregate or component grouping, or sharing objects.

Aggregate Classes interact through messages, which are requests for services from one Class (called a client), to another Class (called a server). A Polymorphism is termed as the ability of two or more Objects to interpret same message differently during execution, depending upon the superclass of the calling Object.

For example, consider a car owned by you as an object. The object is complete in itself and all necessary data (components and specifications) are embedded into the object. The object can be specifically used for the purpose it has been designed. However, there are different objects either having similar data (same model, same company) or different data (Different model, different companies etc.) All these objects belong to class cars. All object cars have common attributes (i.e. steering, gear, break, wheels etc.) that are inherited from class cars (or may be from superclass vehicles). One can modify the object car by keeping basic common attributes and add few more functions to it. (Polymorphism)

There are many programming languages that are used for developing object-oriented systems. To realize the full benefits of using object-oriented programming, it is necessary to employ object-oriented analysis and design approaches. Dealing with objects should permit analysts, developers and programmers to consider larger logical chunks of a system and clarify the programming process. Although it is possible to do object-oriented development using a waterfall model in practice most object-oriented systems are developed with an iterative approach. As a result, in object-oriented processes "Analysis and Design" are often considered at the same time. OOSD being a programming method, use of a particular programming language or a particular programming technique does not imply or require use of a particular software development methodology.

Advantages of OOSD:

- The ability to manage an unrestricted variety of data types
- Provision of a means to model complex relationships
- It has capability to meet the demands of a changing technology and environment

A significant development in OOSD has been the decision by some of the major players in object-oriented development to join forces and merge their individual approaches into a unified approach using the Unified Modelling Language (UML). UML is a general-purpose notational language which helps developers to specify and visualize complex software for large object-oriented projects. This signals a maturation of the object-oriented development approach. While object-orientation is not yet pervasive, it can be accurately said to have entered the computing mainstream.

Applications that use object-oriented technology are:

- Web Applications
- E-Business applications

- CASE for Software Development
- Office Automation for email and work orders
- Artificial Intelligence
- Computer-Aided Manufacturing (CAM) for production and process control

2.6.5 Component Based Development

Component-Based Development is an outgrowth of Object-Oriented Development. Component-Based Development is in fact assembling packages of executable software that make their services available through defined interfaces. These packages also called as enabling pieces of programs are called Objects. These objects are independent of programming languages or operating system. The basic types of Components are:

- **In-Process Client Components:** These components must run from within defined program (called as 'Container') such as a web browser; they cannot run on their own.
- **Stand-Alone Client Components**—Applications (like Microsoft's Excel and Word) that work as service.
- **Stand-Alone Server Components**—Processes running on servers that provide services in standardized way. These are initiated by remote procedure calls or some other kind of network call. Technologies supporting this include Microsoft's Distributed Component Object Model (DCOM), Object Management Group's Common Object Request Broker Architecture (CORBA) and Sun's Java through Remote Method Invocation (RMI).
- **In-Process Server Components:** These components run on servers within containers. Examples include Microsoft's Transaction Server (MTS) and Sun's Organization Java Beans (EJB)

A number of different component models have emerged. E.g. Microsoft's Component Object Model (COM). MTS when combined with COM allows developers to create components that can be distributed in the Windows environment. COM is the basis for ActiveX technologies, with ActiveX Controls being among the most widely used components. Alternative component models include the CORBA Component Model and Sun's EJB.

COM/DCOM, CORBA and RMI are sometimes referred to as Distributed Object Technologies or also termed Middleware. (Middleware is a broad term, but a basic definition is software that provides run-time services where by programs/ objects/ components can interact with one another).

Visual tools are now available for designing and testing Component-Based Applications. Components play a significant role in web-based applications.

Advantages of Component-Based Development are:

- This reduces development time as application system can be assembled from pre-written components and only code for unique parts of the system needs to be developed.
- Improves quality by using pre-written and tested components.
- Allows developers to focus more strongly on business functionality.
- Promotes modularity by encouraging interfaces between discrete units of functionality.
- Simplifies re-use and avoids need to be conversant with procedural or class libraries.
- Combining and allowing reusable code to be distributed in an executable format—i.e., no source is required.
- Reduces development cost as less effort is required for designing and developing the software.
- Supports multiple development environments due to platform independent components.
- Allows a satisfactory compromise between build and buy options i.e. instead of buying a complete solution, it could be possible to purchase only needed components and incorporate these into a customized system.

Disadvantages:

- Attention to software integration should be provided continuously during the development stage.
- If system requirements are poorly defined or the system fails to adequately address business needs, the project will not be successful.

2.6.6 Web-Based Application Development

Web based development has modified client-server architecture and made it more light weight and easy to implement. It has eliminated the need to implement client module on end-user's desktop, and is delivered via internet-based technologies. User need to know URL to access the application and if need be the same is delivered on users' desktop or executed from web server. The technology can be deployed within organization also. For example, many organizations have implemented internal user services using intranet portal, which essentially uses internet (web) based technologies.

Historically, software written in one language on a particular platform has used a dedicated Application Programming Interface (API). The use of specialized APIs has caused difficulties in integrating software modules across platforms. Component Based Technologies such as CORBA and COM that use Remote Procedure Calls (RPCs) have been developed to allow

real-time integration of code across platforms. However, using these RPC approaches for different APIs still remains complex. Web-based application development is designed to further facilitate and standardize code module and program integration.

Web-based application development enables users to avoid the need to perform redundant computing tasks with redundant code. For example, installing client on all users after making changes or change of address notification from a customer need not be updated separately in multiple databases. For example, entering and maintaining same data in contact management, accounts receivable etc. Web application development though is different than traditional developments (e.g. users test and approve the development work), but the risks of application development remain the same.

With web-based application development, an XML language known as Simple Object Access Protocol (SOAP) is used to define APIs. SOAP will work with any operating system and programming language that understands XML. SOAP is simpler than using the more complex RPC-based approach, with the advantage that modules are coupled loosely so that a change to one component does not normally require changes to other components. The second key component of web development is the Web Services Description Language (WSDL), which is also based on XML. WSDL is used to identify the SOAP specification that is to be used for the code module API and the formats of the SOAP messages used for input and output to the code module. The WSDL is also used to identify the particular web service accessible via a corporate intranet or across the Internet by being published to a relevant intranet or Internet web server.

2.7 Selection of SDLC Model

Assess the needs of Stakeholders

We must study the business domain, stakeholders' concerns and requirements, business priorities, our technical capability and ability, and technology constraints to be able to choose the right SDLC against their selection criteria.

Define the criteria

Some of the selection criteria or arguments that you may use to select an SDLC are:

- Is the SDLC suitable for the size of our team and their skills?
- Is the SDLC suitable for the selected technology we use for implementing the solution?
- Is the SDLC suitable for client and stakeholders' concerns and priorities?
- Is the SDLC suitable for the geographical situation (distributed team)?
- Is the SDLC suitable for the size and complexity of our software?
- Is the SDLC suitable for the type of projects we do?

- Is the SDLC suitable for our software engineering capability?
- Is the SDLC suitable for the project risk and quality insurance?

2.8 New Development Iterative Models- Prototype, Spiral, Rapid & Agile etc.

2.8.1 Prototyping Methodology

The traditional approach sometimes may take long time to analyze, design and implement a system. More so, many a times we know a little about the system until and unless we go through its working phases, which are not available. In order to avoid such bottlenecks and overcome the issues, organizations are increasingly using prototyping techniques to develop smaller systems such as DSS, MIS and Expert systems. The goal of prototyping approach is to develop a small or pilot version called a prototype of part or all of a system. A prototype may be a usable system or system component that is built quickly and at a lesser cost, and with the intention of modifying/replicating/expanding or even replacing it by a full-scale and fully operational system. As users work with the prototype, they learn about the system criticalities and make suggestions about the ways to manage it. These suggestions are then incorporated to improve the prototype, which is also used and evaluated. Finally, when a prototype is developed that satisfies all user requirements, either it is refined and turned into the final system or it is scrapped. If it is scrapped, the knowledge gained from building the prototype is used to develop the real system.

Prototyping can be viewed as a series of four steps, symbolically depicted in Fig. 2.4 wherein implementation and maintenance phases followed by full-blown developments take place once the prototype model is tested and found to be meeting users' requirements.

Generic Phases of Model

- **Identify Information System Requirements:** In traditional approach, the system requirements are to be identified before the development process starts. However, under prototype approach, the design team needs only fundamental system requirements to build the initial prototype, the process of determining them can be less formal and time-consuming than when performing traditional systems analysis.
- **Develop the Initial Prototype:** The designers create an initial base model and give little or no consideration to internal controls, but instead emphasize system characteristics such as simplicity, flexibility, and ease of use. These characteristics enable users to interact with tentative versions of data entry display screens, menus, input prompts, and source documents. The users also need to be able to respond to system prompts, make inquiries of the information system, judge response times of the system, and issue commands.

- **Test and Revise:** After finishing the initial prototype, the designers first demonstrate the model to users and then give it to them to experiment and ask users to record their likes and dislikes about the system and recommend changes. Using this feedback, the design team modifies the prototype as necessary and then re-submits the revised model to system users for re-evaluation. Thus, iterative process of modification and re-evaluation continues until the users are satisfied.
- **Obtain User Signoff of the Approved Prototype:** Users formally approve the final version of the prototype, which commits them to the current design and establishes a contractual obligation about what the system will, and will not do or provide. Prototyping is not commonly used for developing traditional MIS and batch processing type of applications such as accounts receivable, accounts payable, payroll, or inventory management, where the inputs, processing, and outputs are well known and clearly defined.

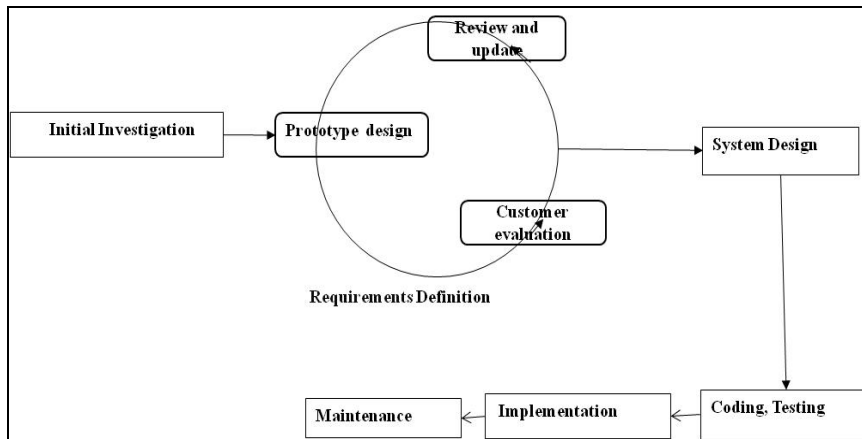


Fig. 2.4: Prototyping Model

Strengths:

- It improves both user participation in system development and communication among project stakeholders.
- It is especially useful for resolving unclear objectives and requirements; developing and validating user requirements; experimenting with or comparing various design solutions, or investigating both performance and the human computer interface.
- Potential exists for exploiting knowledge gained in an early iteration as later iterations are developed.
- It helps to easily identify confusing or difficult functions and missing functionality.
- It enables to generate specifications for a production application.
- It encourages innovation and flexible designs.

- It provides for quick implementation of an incomplete, but functional application.
- It typically results in a better definition of these users' needs and requirements than does the traditional systems development approach.
- A very short time period is normally required to develop and start experimenting with a prototype. This short time period allows system users to immediately evaluate proposed system changes.
- Since system users experiment with each version of the prototype through an interactive process, errors are hopefully detected and eliminated early in the developmental process. As a result, the information system ultimately implemented should be more reliable and less costly to develop than when the traditional systems development approach is employed.

Weaknesses:

- Approval process and control are not formal.
- Incomplete or inadequate problem analysis may occur whereby only the most obvious and superficial needs will be addressed, resulting in current inefficient practices being easily built into the new system.
- Requirements may frequently change significantly.
- Identification of non-functional elements is difficult to document.
- Designers may prototype too quickly, without sufficient upfront user needs analysis, resulting in an inflexible design with narrow focus that limits future system potential.
- Prototype may not have sufficient checks and balances incorporated.
- Prototyping can only be successful if the system users are willing to devote significant time in experimenting with the prototype and provide the system developers with change suggestions. The users may not be able or willing to spend the amount of time required under the prototyping approach.
- The interactive process of prototyping causes the prototype to be experimented with quite extensively. Because of this, the system developers are frequently tempted to minimize the testing and documentation process of the ultimately approved information system. Inadequate testing can make the approved system error-prone, and inadequate documentation makes this system difficult to maintain.
- Prototyping may cause behavioural problems with system users. These problems include dissatisfaction by users if system developers are unable to meet all user demands for improvements as well as dissatisfaction and impatience by users when they have to go through too many interactions of the prototype.
- In spite of above listed weaknesses, to some extent, systems analysis and development has been greatly improved by the introduction of prototyping. Prototyping enables the

user to take an active part in the systems design, with the analyst acting in an advisory role. Prototyping makes use of the expertise of both the user and the analyst, thus ensuring better analysis and design, and prototyping is a crucial tool in that process.

Prototype has one major drawback. Many-a-time users do not realize that prototype is not actual system or code but is just a model. Users may think that the system is ready. Whereas actual development starts only after the prototype is approved. Hence, the actual system may require time before it is ready for implementation and use. In the meantime, users may get restless and wonder why there is so much delay.

2.8.2 Spiral Model

The Spiral model is a repetitive software development process combining elements of both design and prototyping within each of the iterations. It combines the features of the prototyping model and the waterfall model (given in Fig. 2.5). Initially spiral model was intended for large, expensive and complicated projects like Game development because of size and constantly shifting goals of large projects. Spiral model when defined was considered as the best model and further models were developed using spiral models.

Key characteristics

- Spiral model is an iterative model where each iteration helps in optimizing the intended solution.
- The new system requirements are defined in as much detail as possible. This usually involves interviewing a number of users representing all the external or internal users and other aspects of the existing system.
- A preliminary design is created for the new system during initial iterations. This phase is the most important part of “Spiral Model” in which all possible alternatives that can help in developing a cost-effective project are analysed and strategies are decided to use them. This phase has been added specially in order to identify and resolve all the possible risks in the project development. If risks indicate any kind of uncertainty in requirements, prototyping may be used to proceed with the available data and find out possible solution in order to deal with the potential changes in the requirements.
- A first prototype of the new system is constructed from the preliminary design during first iteration. This is usually a scaled-down system, and represents an approximation of the characteristics of the final product.
- A second prototype is evolved during next iteration by a fourfold procedure by evaluating the first prototype in terms of its strengths, weaknesses, and risks; defining the requirements of the second prototype; planning and designing the second prototype; and constructing and testing the second prototype.

Strengths:

- Enhances the risk avoidance.
- Useful in helping for optimal development of a given software iteration based on project risk.
- Incorporates Waterfall, Prototyping, and Incremental methodologies as special cases in the framework, and provide guidance as to which combination of these model's best fits a given software iteration, based upon the type of project risk. For example, a project with low risk of not meeting user requirements but high risk of missing budget or schedule targets would essentially follow a linear Waterfall approach for a given software iteration. Conversely, if the risk factors were reversed, the Spiral methodology could yield an iterative prototyping approach.

Weaknesses:

- It is challenging to determine the exact composition of development methodologies to use for each of the iterations around the Spiral.
- A skilled and experienced Project Manager is required to determine how to apply it to any given project.
- Sometimes there are no firm deadlines, cycles continue till requirements are clearly identified. Hence has an inherent risk of not meeting budget or schedule.

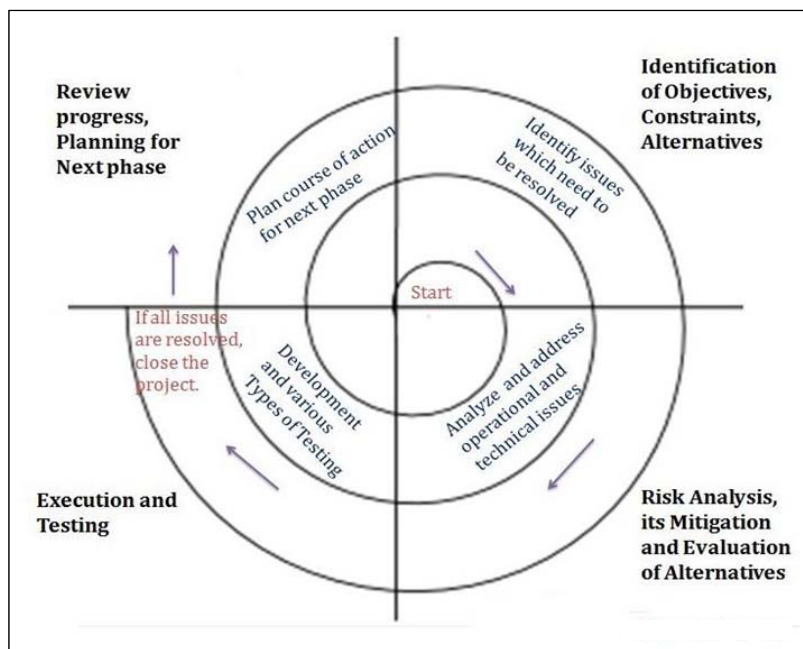


Fig. 2.5: Spiral Model

2.8.3 Rapid Application Development (RAD)

RAD refers to a type of software development methodology, which uses minimal planning in favour of rapid prototyping. (Figure 2.6) The planning of software developed using RAD is interleaved with writing the software itself. The lack of extensive pre-planning generally allows software to be written much faster, and makes it easier to change requirements. The key features are:

- Key objective is fast development and delivery of a high-quality system at a relatively low investment cost,
- Attempts to reduce inherent project risk by breaking a project into smaller segments and providing more ease-of-change during the development process.
- Aims to produce high quality systems quickly, primarily through the use of Iterative Prototyping (at any stage of development), active user involvement, and computerized development tools like Graphical User Interface (GUI) builders, Computer Aided Software Engineering (CASE) tools, Database Management Systems (DBMS), Fourth Generation Programming Languages, Code Generators and Object-Oriented Techniques.
- Key emphasis is on fulfilling the business need while technological or engineering excellence is of lesser importance.
- Project control involves prioritizing development and defining delivery deadlines or "time boxes." If the project starts to slip, emphasis is on reducing requirements to fit the time box, not in increasing the deadline.
- Generally, includes Joint Application Development (JAD), where users are intensely involved in system design, either through consensus building in structured workshops, or through electronically facilitated interaction.
- Active user involvement is imperative.
- Iteratively produces production software, as opposed to a throwaway prototype.
- Produces documentation necessary to facilitate future development and maintenance.
- Standard systems analysis and design techniques can be fitted into this framework.

Strengths:

- The operational version of an application is available much earlier than with Waterfall, Incremental, or Spiral frameworks.
- Because RAD produces systems more quickly and to a business focus, this approach tends to produce systems at lower cost.

- Quick initial reviews are possible.
- Constant integration isolates problems and encourages customer feedback.
- It holds a great level of commitment from stakeholders, both business and technical, than Waterfall, Incremental, or Spiral frameworks. Users are seen as gaining more of a sense of ownership of a system, while developer are seen as gaining more satisfaction from producing successful systems quickly.
- It concentrates on essential system elements from user viewpoint.
- It provides for the ability to rapidly change system design as demanded by users.
- It leads to a tighter fit between user requirements and system specifications.

Weaknesses:

- Fast speed and lower cost may affect adversely the system quality.
- The project may end up with more requirements than needed (gold-plating).
- Potential for feature creep where more and more features are added to the system during development.
- It may lead to inconsistent designs within and across systems.
- It may call for violation of programming standards related to inconsistent naming conventions and inconsistent documentation,
- It may call for lack of attention to later system administration needs built into system.
- Formal reviews and audits are more difficult to implement than for a complete system.
- Tendency for difficult problems to be pushed to the future to demonstrate early success to management.
- As some modules are completed much earlier than others, well-defined interfaces are required.

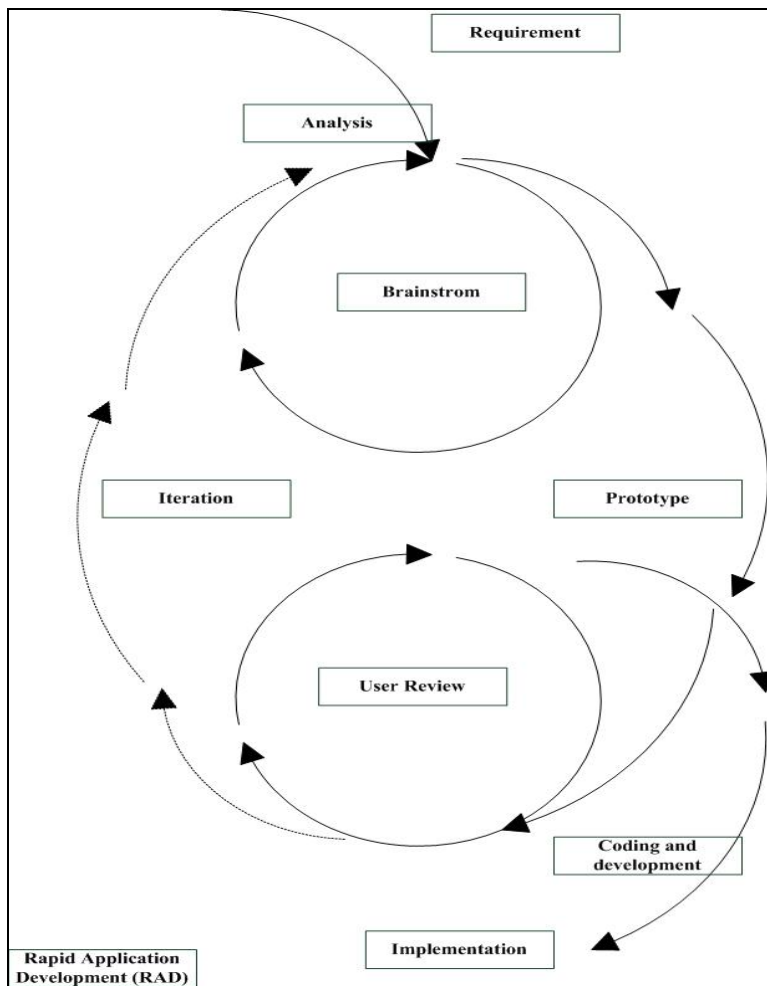


Figure 2.6: RAD

2.8.4 Agile Software Development Methodology

The term “Agile Development” refers to a family of similar development processes that adopt a non-traditional way of developing complex systems. The term “Agile” refers to characteristic of processes that are designed to flexibly handle changes to the systems being developed. Scrum is the first project management approach that fits well with other agile techniques. Other agile processes such as Extreme Programming (XP), Crystal, Adaptive Software Development, Feature Driven Development and Dynamic Systems Development Method have since emerged.

Key Characteristics of Agile processes

- Use of small, time-boxed subprojects or iterations where each iteration forms the basis for planning next iteration.
- Re-planning the project at the end of each iteration (referred to as a “Sprint” in Scrum), including re-prioritizing requirements, identifying any new requirements and determining that the delivered functionality should be implemented within which release
- Relatively greater reliance, compared to traditional methods, on the knowledge in people’s heads (tacit knowledge), as opposed to external knowledge that is captured in project documentation
- A heavy influence on mechanisms to effectively disseminate tacit knowledge and promote teamwork. Therefore, teams are kept small in size, comprise both business and technical representatives, and are located physically together. Team meetings to verbally discuss progress and issues that occur daily, but with strict time limits.
- At least some of the agile methods stipulate pair-wise programming (two persons code the same part of the system) as a means of sharing knowledge and as a quality check.
- A change in the role of the Project Manager, from one primarily concerned with planning the project, allocating tasks and monitoring progress to that of a facilitator and advocate. Responsibility for planning and control is delegated to the team members.

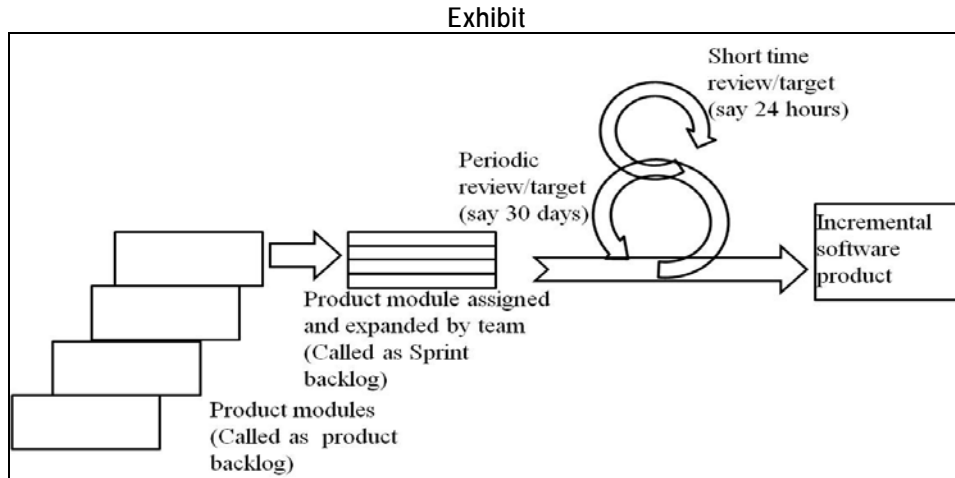


Figure 2.7: Agile (Sprint) review cycle

The agile methodology may be considered as *iterative and incremental* development, where requirements and solutions evolve through collaboration between self-organizing, cross-functional teams. It promotes adaptive planning, evolutionary development and delivery; time

boxed iterative approach and encourages rapid and flexible response to change. It is a conceptual framework that promotes foreseen interactions throughout the development life cycle.

Key features of agile methodologies

- Customer satisfaction by rapid delivery of useful software;
- Welcome changing requirements, even late in development;
- Working software is delivered frequently (weeks rather than months);
- Working software is the principal measure of progress;
- Sustainable development, able to maintain a constant pace;
- Close and regular interaction between business representatives and developers;
- Face-to-face conversation is the best form of communication (co-location);
- Projects are built around motivated individuals, who should be trusted;
- Continuous attention to technical excellence and good design;
- Simplicity; Self-organizing teams; and
- Regular adaptation to changing circumstances.

Strengths:

- Agile methodology has the concept of an adaptive team, which enables to respond to the changing requirements.
- The team does not have to invest time and efforts and finally find that by the time they delivered the product, the requirement of the customer has changed.
- Face to face communication and continuous inputs from customer representative leaves a little space for guesswork.
- The documentation is crisp and to the point to save time.
- In general, the end result is of high-quality software in least possible duration leading finally to a satisfied customer.

Weaknesses:

- In case of some software deliverables, especially the large ones, it is difficult to assess the efforts required at the beginning of the System Development life cycle.
- There is lack of emphasis on necessary designing and documentation due to time management. As a result, documentation is generally left out or remains incomplete.

- In Agile methodology, there is a possibility of increasing potential threats to the knowledge transfer and business continuity due to verbal communication and weak documentation.
- In Agile methodology there is no long-term planning. Also, the approach to the architecture is lightweight. Hence, it requires more re-work.
- The project can easily go off the track if the customer representative is not having clarity about the requirements and final deliverables.
- Agile methodology lacks the attention to external integration.

2.8.5 DevOps

Historically, the concept of DevOps came into existence for developing a culture of collaboration between the teams. DevOps refers to the integration of development and operations processes to eliminate conflicts and barriers. This integration can create a great deal of benefits, but it can also create new risk. Decisions to adopt DevOps should be made based on factors such as an organization's climate, risk tolerance and culture and on the scope of the development project. Because DevOps changes the environment and often impacts an organization's control environment and accepted level of risk, an **IS Auditor** should ensure that there is a proper separation of duties.

DevOps combines the concepts of agile development, agile infrastructure and flexible operations. It requires a bridge of communication between software development and operations and the application of agile principles to all functions that support the Software Development Life Cycle. Implementing DevOps processes can be done in a logical and systematic manner and used to enhance the maturity of software development.

2.8.6 DevSecOps

DevSecOps means building security into app development from end to end. The adoption DevSecOps is often closely associated with the adoption of Agile. DevSecOps uses two distinctive concepts: (1) the confluence of software development, Information Security and IT operations groups and (2) the use of automation in those activities.

An organization should consider the following controls when embracing a DevOps development approach:

- Automated Software Scanning
- Automated Vulnerability Scanning
- Web Application Firewall
- Developer Application Security Training

- Software Dependency Management
- Access and Activity Logging
- Documented Policies and Procedures
- Application Performance Management
- Asset Management and inventorying
- Continuous Auditing and/or Monitoring
- Encrypt Data between Apps and Services

2.9 Secure SDLC

Earlier security was an afterthought for SDLC; normally developers used to check the security related aspects through penetration testing, which requires a lot of rework. For example, if a security related vulnerability, bug or flaw is detected after development then correction of the same will require re-examining all the aspects starting from requirements till coding. This entire exercise will increase the cost and efforts of the project, which sometimes may create a typical situation both for the client and development company. To overcome this issue, latest research studies suggest that the security should be taken into account right from the beginning in the SDLC. Information security trends indicate that embedding security within application development helps in addressing various issues. For example, when multiple users are expected to access application hosted at central location from different nodes, the application should be able to provide access depending upon the function the specific users has to perform. This requires designing role definition and assigning various roles to different users according to their functionality.

Another example can be in case the application is developed using web-based technologies and users are expected to access it using different browsers (like internet explorer, Google chrome etc.), application may not depend upon users to secure their browsers, but embed security within application. In case the application is hosted on internet, it is subject to various application level attacks (like OWASP top 10 2017) that need to be closed by adopting secure development and coding practices.

The following table describes the additional activities that need to be added to the traditional SDLC phases to make it Secure SDLC.

Table 2.1: Security steps in various phases of SDLC

SDLC Phase	Security Steps
Requirement Definition	<p>To identify security requirements including compliance for privacy and data loss.</p> <p>To determine risks associated with security and prepare mitigation plan.</p> <p>To train users on identification and fixing of security bugs.</p>
Design Phase	<p>To ensure security requirements are considered during design phase e.g. access controls for privacy sensitive data.</p> <p>To identify possible attacks and design controls e.g. implementing least privilege principle for sensitive data, and apply layered principle for modules.</p>
Development Phase	<p>To develop and implement security coding practices such as input data validation and avoiding complex coding.</p> <p>To train developers on security coding practices.</p>
Testing Phase	<p>To review code for compliance of secure coding practices.</p> <p>To develop test cases for security requirement testing.</p> <p>To ensure security requirements are tested during testing.</p> <p>To test application for identified attacks.</p>
Implementation Phase	<p>To analyze all functions and interfaces are secured.</p> <p>To perform security scan of application after implementation.</p>
Maintenance Phase	<p>To monitor for vulnerabilities on a continuous basis,</p> <p>To issue the patches for fixing the reported vulnerabilities, accordingly,</p> <p>To evaluate the effectiveness of countermeasures periodically.</p>

2.10 Summary

SDLC is an essential aspect of automating business processes using Information Technology. It has been evolving with changing technology and global proliferation of computers. Today's business largely depends on IT and any problem faced has multi-fold repercussions. Controlling SDLC process helps organizations in mitigating risks associated with implementation and use of IT. An IS Auditor must be aware of phases and key steps of each

of the SDLC phases. There are various models and methods. An IS auditor, while auditing SDLC process is not required to be an expert in all technologies but should always focus on associated risks, assessment of these risks and assess whether the implemented solutions are as per the expected business objectives.

2.11 Questions

1. SDLC primarily refers to the process of:
 - A. Developing IT based solution to improve business service delivery.
 - B. Acquiring upgraded version of hardware for existing applications.
 - C. Redesigning network infrastructure as per service provider's needs.
 - D. Understanding expectations of business managers from technology.
2. Organizations should adopt programming/coding standards mainly because, it:
 - A. Is a requirement for programming using High Level Languages?
 - B. Helps in maintaining and updating System Documentation.
 - C. Is required for Security and Quality Assurance function of SDLC.
 - D. Has been globally accepted practice by large organizations.
3. An organization decided to purchase a configurable application product instead of developing in-house. Outcome of which of the following SDLC phase helped organization in this decision?
 - A. Requirement Definition
 - B. Feasibility Study
 - C. System Analysis
 - D. Development Phase
4. In which of the following phases of SDLC, controls for security must be considered FIRST?
 - A. Requirement Definition
 - B. Feasibility Study
 - C. System Design
 - D. Implementation

5. IS Auditor has been part of SDLC project team. Which of the following situation does not prevent IS Auditor from performing post implementation review? The IS Auditor has:
 - A. Designed the Security Controls.
 - B. Implemented Security Controls.
 - C. Selected Security Controls.
 - D. Developed Integrated Test facility.
6. An organization has implemented an IT based solution to support business function. Which of the following situation shall indicate the need to initiate SDLC project?
 - A. Vendor has launched a new hardware which is faster.
 - B. Organizations has unused surplus budget for IT.
 - C. Regulators have requested additional reports from business.
 - D. Competitor has launched an efficient IT based service.
7. A "Go or No Go" decision for SDLC project is primarily based on:
 - A. Feasibility Study
 - B. Business Case
 - C. Budget Provision
 - D. Market Situation
8. Which of the following is the primary reason for organization to outsource the SDLC project? Non-availability of:
 - A. Skilled Resources
 - B. Budgetary Approvals
 - C. Security Processes
 - D. Infrastructure
9. Which of the following is an example of addressing social feasibility issue in SDLC project?
 - A. Organization decides to use existing infrastructure.
 - B. Beta version of the application is made available to users.
 - C. Configuration of purchased software requires more cost.
 - D. Allowing employees to access social media sites.

10. Which of the following is not an indicator to assess benefit realization for internal application software developed in-house?
- A. Increase in number of customers because of new application.
 - B. Decrease in audit findings related to regulatory non-compliance.
 - C. Reduced number of virus attacks after implementing new software.
 - D. Increase in productivity of employees after implementation.

2.12 Answers and Explanations

1. A is correct answer. SDLC primarily focuses on identifying IT based solution to improve business processes delivering services to customers. Other activities may be part of SDLC however, these are IT projects not SDLC projects.
2. C is correct answer. Adopting coding standards helps organization in ensuring quality of coding and in minimizing the errors. It also helps in reducing obvious errors which may lead to vulnerabilities in application. A is not true since it is required for all languages; B is partially true but is not main reason. D is not main reason.
3. B is the correct answer. Make or buy decision is the outcome of feasibility study where technical, economical and social feasibilities are considered. Option A is a statement that indicates what a system needs to do in order to provide a capability. Options C and D are the phases of developing a software.
4. A is the correct answer. Security requirements must be considered during requirement definition. Option B is a phase in which technical, economical and social feasibilities are considered. Option C is the phase during which, the nature of controls to be implemented for security must be considered first. This will ensure that necessary security controls are built while developing application.
5. D is the correct answer. Active role of IS Auditor in design and development of controls affects the independence. Hence, IS Auditor cannot perform review or audit of the application system. However, developing integrated test facility within the application is not a control, but a facility to be used by auditors in future. Hence, this does not impact independence of IS auditor. Options A, B and C affect independence of an IS Auditor.
6. D is correct answer. When a competitor launches new IT based efficient service, it becomes necessary for management to consider the impact in market place and in order to remain in competition organization should provide similar or better services. Option A and C may not require SDLC since it can be adopted with change management process. B may help in deciding for D, but is not the reason for initiating SDLC project.

7. B is the correct answer. Business case is a document that narrates all aspect including benefit realization, cost and effort estimates, outcome of feasibility study, available budget. That helps management in decision on the need of the SDLC project. Rest are secondary aspects.
8. A is correct answer. Non availability of skilled resources required for application development is primary reason for outsourcing the SDLC project. Other reasons can be addressed. i.e. (B) budget can be made available; (C) security processes can be established. (D) Infrastructure can be acquired, depending upon design of new application and hence it is not a reason.
9. B is the correct answer. In order to ensure the acceptability by users, beta version of solution is made available to users. Based on feedback changes are made so that the solution can be socialized. Option A addresses technical feasibility, Option C addresses economic feasibility. Option D addresses IT policy that has nothing to do with SDLC.
10. C is the correct answer. Since the application is for internal use and developed in house it has nothing to do with reduction in virus attacks. This can be benefit realization for anti-virus solution.

Software Testing and Implementation

Learning Objectives

This chapter will give you a basic understanding on software testing, its importance, strategies, types, methods, levels, and other related terminologies.

3.1 Introduction

The success of information systems depends upon the quality of software that supports the system. Testing of software before deploying in production to ensure it delivers as per requirements is most essential aspect of quality. This is apart from documentation, compliance with coding standards, version control discipline and user training.

Testing is the process of evaluating a system or its component(s) with the intent to find whether it satisfies the specified requirements or not. In simple words, testing is executing a system in order to identify any gaps, errors, or missing requirements in contrary to the actual requirements.

According to ANSI/IEEE 1059 standard, Testing can be defined as - A process of analyzing a software item to detect the differences between existing and required conditions (that is defects/errors/bugs) and to evaluate the features of the software item.

3.2 Importance of Software Testing

The programmer might have made mistakes that need to be tested. Some mistakes come from bad assumptions and blind spots, which should be tested other persons. There are several reasons, which clearly indicate the importance of software testing. These are:

- Software testing is really required to point out the defects and errors that were made during the development phase.
- It's essential since it makes sure that the user finds the software reliable and their satisfaction in the application is maintained.
- It is very important to ensure the Quality of the product. Quality product delivered to the users helps in gaining their confidence.
- Software testing is important in order to provide the facilities to the customers like the delivery of high-quality product or software application which requires lower maintenance cost and hence results into more accurate, consistent and reliable outputs.

- Software testing is required for an effective performance of application or product.
- It's important to ensure that the application should not result into any failures because it can be very expensive in the future or in the later stages of the development.
- Proper testing ensures that bugs and issues are detected early in the life cycle of the product or application.
- Users are not inclined to use software that has bugs. They may not adopt a software if they are not satisfied with the stability of the application.

3.3 Methods of Software Testing

There are different methods that can be used for software testing. This chapter briefly describes the methods available.

3.3.1 Black-Box Testing

The technique of testing without having any knowledge of the interior workings of the application is called Black-Box testing. The tester is oblivious to the system architecture and does not have access to the source code. Typically, while performing a Black-Box test, a tester will interact with the system's user interface by providing inputs and examining outputs without knowing how and where the inputs are worked upon.

The following table lists the advantages and disadvantages of black-box testing.

Advantages	Disadvantages
Well suited and efficient for large code segments.	Limited coverage, since only a selected number of test scenarios is actually performed.
Code access is not required.	Inefficient testing, due to the fact that the tester only has limited knowledge about an application.
Clearly separates user's perspective from the developer's perspective through visibly defined roles.	Blind coverage, since the tester cannot target specific code segments or error prone areas.
Large numbers of moderately skilled testers can test the application with no knowledge of implementation, programming language, or operating systems.	The test cases are difficult to design.

3.3.2 White-Box Testing

White-Box testing is the detailed investigation of internal logic and structure of the code. White-Box testing is also called Glass testing or Open-Box testing. In order to perform White-Box testing on an application, a tester needs to know the internal workings of the code.

The tester needs to have a look inside the source code and find out which unit/chunk of the code is behaving inappropriately.

The following table lists the advantages and disadvantages of White-Box testing.

Advantages	Disadvantages
As the tester has knowledge of the source code, it becomes very easy to find out which type of data can help in testing the application effectively.	Due to the fact that a skilled tester is needed to perform white-box testing, the costs are increased.
It helps in optimizing the code.	Sometimes it is impossible to look into every nook and corner to find out hidden errors that may create problems, as many paths will go untested.
Extra lines of code can be removed which can bring in hidden defects.	It is difficult to maintain white-box testing, as it requires specialized tools like code analyzers and debugging tools.
Due to the tester's knowledge about the code, maximum coverage is attained during test scenario writing.	

3.3.3 Grey-Box Testing

Grey-Box testing is a technique to test the application with having a limited knowledge of the internal workings of an application. In software testing, the phrase “The more you know, the better” carries a lot of weight while testing an application.

Mastering the domain of a system always gives the tester an edge over someone with limited domain knowledge. Unlike Black-Box testing, where the tester only tests the application's user interface; in Grey-Box testing, the tester has access to design documents and the database. Having this knowledge, a tester can prepare better test data and test scenarios while making a test plan.

Advantages	Disadvantages
Offers combined benefits of Black-Box and White-Box testing wherever possible.	Since the access to source code is not available, the ability to go over the code and test coverage is limited.
Grey-Box testers don't rely on the source code; instead they rely on interface definition and functional specifications.	The tests can be redundant if the software designer has already run a test case.
Based on the limited information available, a Grey-Box tester can design excellent test scenarios especially around communication protocols and data type handling.	Testing every possible input stream is unrealistic because it would take an unreasonable amount of time. As a result, many program paths will go untested.
The test is done from the point of view of the user and not the designer.	

3.3.4 A Comparison of Testing Methods

The following table lists the points that differentiate Black-Box testing, Grey-Box testing, and White-Box testing.

Black-Box Testing	Grey-Box Testing	White-Box Testing
The internal workings of an application need not be known.	The tester has limited knowledge of the internal workings of the application.	Tester has full knowledge of the internal workings of the application.
Also known as Closed-Box testing, Data-Driven testing, or Functional testing.	Also known as Translucent testing, as the tester has limited knowledge of the insides of the application.	Also known as Clear-Box testing, Structural testing, or Code-Based testing.
Performed by end-users and also by testers and developers.	Performed by end-users and also by testers and developers.	Normally done by testers and developers.
Testing is based on external expectations - Internal behavior of the application is unknown.	Testing is done on the basis of high-level database diagrams and data flow diagrams.	Internal workings are fully known and the tester can design test data accordingly.
It is exhaustive and the least time-consuming.	Partly time-consuming and exhaustive.	The most exhaustive and time-consuming type of testing.

Not suited for algorithm testing.	Not suited for algorithm testing.	Suited for algorithm testing.
This can only be done by trial-and-error method.	Data domains and internal boundaries can be tested, if known.	Data domains and internal boundaries can be better tested.

3.4 Levels of Testing

There are different levels during the process of testing. In this chapter, a brief description is provided about these levels.

Levels of testing include different methodologies that can be used while conducting software testing. The main levels of software testing are –

- Functional Testing
- Non-Functional Testing

3.4.1 Functional Testing

This is a type of Black-Box testing that is based on the specifications of the software that is to be tested. The application is tested by providing input and then the results are examined that need to conform to the functionality it was intended for. Functional testing of software is conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements.

There are five steps that are involved while testing an application for functionality.

Steps	Description
I	The determination of the functionality that the intended application is meant to perform.
II	The creation of test data based on the specifications of the application.
III	The output based on the test data and the specifications of the application.
IV	The writing of test scenarios and the execution of test cases.
V	The comparison of actual and expected results based on the executed test cases.

An effective testing practice will see the above steps applied to the testing policies of every organization and hence it will make sure that the organization maintains the strictest of standards when it comes to software quality. Commonly used functional testing types are; Unit

testing, Integration testing, Smoke testing, Sanity testing, System testing, Regression testing, Acceptance testing (Alpha testing, and Beta testing), and End to End testing.

3.4.2 Non-Functional Testing

This section is based upon testing an application from its Non-Functional attributes. Non-Functional testing involves testing a software from the requirements which are non-functional in nature but important such as performance, security, user interface, etc.

Some of the important and commonly used non-functional testing types are Performance testing, Load testing, Stress testing, Usability testing, Security testing, and Portability testing.

3.5 Strategies of Software Testing

3.5.1 What is a Test Strategy?

Test strategy is a guideline to be followed to achieve the test objective and execution of test types mentioned in the testing plan. It deals with test objective, test environment, test approach, automation tools and strategy, contingency plan, and risk analysis.

A Test Strategy is a plan for defining the testing approach as to how testing would be carried out. Test approach has two techniques:

- Proactive - An approach in which the test design process is initiated as early as possible in order to find and fix the defects before the build is created.
- Reactive - An approach in which the testing is not started until after design and coding are completed.

3.5.2 Different Test approaches

There are many strategies that a project can adopt depending on the context and some of them are:

- Dynamic and Heuristic approaches
- Consultative approaches
- Model-Based approach that uses statistical information about failure rates.
- Approaches based on Risk-Based testing where the entire development takes place based on the risk
- Methodical approaches which is based on failures.
- Standard-Compliant approach specified by industry-specific standards.

3.5.3 Factors to be considered

- Risks of product or risk of failure or the environment and the company
- Expertise and experience of the people in the proposed tools and techniques.
- Regulatory and legal aspects, such as external and internal regulations of the development process
- The nature of the product and the domain

3.6. Types of Software Testing

3.6.1 Unit Testing

In computer programming, Unit Testing is a verification and validation method in which a programmer tests whether individual units of source code are fit for use. A Unit is the smallest functional part of an application often called as Module. It can be an individual program, function, procedure, or may belong to a base/super class, abstract class or derived/child class.

Unit Tests are typically written based on requirement specifications and run by testing professionals or software developers to ensure that code meets these requirements and behaves as intended. The goal of Unit Testing is to isolate each component of the program and show that they are correct. A Unit Test provides a strict, written contract that the piece of code must satisfy.

There are five categories of tests typically performed on a program unit. Such typical tests are described as follows:

1. **Functional Tests:** Functional Tests check 'whether programs do what they are supposed to do. The test plan specifies operating conditions, input values, and expected results, and as per this plan, programmer checks by inputting the values to see whether the actual results and expected results match. These test data values are prepared in advance for all possible permutations the data can acquire during live run. This can have two types:

- (a) **Positive Test:** Where tester collects the expected values, the data can possess. Sometimes tester may use sanitized live data for testing.
- (b) **Negative Test:** Where tester provides value sets that data should not possess anytime. Here the program should flash the error with suitable message.

For example, if data field is used to store amount and can acquire a value between 0 and 1 crore, positive test should provide expected results and negative test should flash error if values are beyond the specified range.

2. **Performance Tests:** Performance Tests are designed to verify the expected performance criteria of program.

There are different performance parameters like response time (time required to receive input and deliver confirmation), execution time (processing of single data value should be less than 100 microseconds), throughput (1000 values must be processed in one second), primary (RAM/CPU) and secondary memory (Storage) utilization and rate of traffic flow on data channels and communication links (number of messages per second).

3. **Stress Tests:** Stress Testing is a form of testing that is used to determine the stability of a given system or entity. It involves testing beyond normal operational capacity, often to a breaking point, in order to observe the results. These tests are designed to overload a program in various ways. The purpose of a Stress Test is to determine the limitations of the program. (For example, if access to web application is expected to be 10000 hits per second, whether the program can stand this load. Further, how does it behave when load exceeds. In another example, during a sort/search operation, available memory can be reduced to find out whether the program is able to handle the situation.).

4. **Structural Tests:** Structural Tests are concerned with examining the internal processing logic of a software system. Particularly when the program is expected to behave differently depending upon value of data set. Programmer may code for known values and might forget to code for unknown values where program might misbehave. For example, tax calculation where depending upon value different rates is applied or if division operation is involved and data set gets value of zero, program may terminate abruptly or go in loop without response.

5. **Parallel Tests:** These are applicable during change management or reengineering where the same test data is used in the new and old system and the output results are then compared.

3.6.2 Static Testing

Static Test analysis is conducted on source programs and do not normally require executions in operating conditions. Typical static analysis techniques include the following:

- **Desk Check:** This is done by the programmer to check the logical syntax errors, and deviation from coding standards. As name suggests programmer uses paper and pen to verify the logic of code by jotting down values of data sets and thinking like computer to arrive at possible values.
- **Structured Walk-through:** Desk check performed with team or peers who scan through the text of the program and try to uncover errors.
- **Code Inspection:** The program is reviewed by a formal committee. Review is done with formal checklists.

3.6.3 Load Testing

It is a process of testing the behavior of a software by applying maximum load in terms of software accessing and manipulating large input data. It can be done at both normal and peak load conditions. This type of testing identifies the maximum capacity of software and its behavior at peak time.

Most of the time, Load Testing is performed with the help of automated tools such as Load Runner, App Loader, IBM Rational Performance Tester, Apache JMeter, Silk Performer, Visual Studio Load Test, etc.

Virtual Users (V Users) are defined in the automated testing tool and the script is executed to verify the load testing for the software. The number of users can be increased or decreased concurrently or incrementally based upon the requirements.

3.6.4 Usability Testing

Usability Testing is a Black-Box Technique and is used to identify any error(s) and improvements in the software by observing the users through their usage and operation.

According to Nielsen, Usability can be defined in terms of five factors, i.e. efficiency of use, learning-ability, memory-ability, errors/safety, and satisfaction. According to him, the Usability of a product will be good and the system is Usable if it possesses the above factors.

Nigel Bevan and Macleod considered that Usability is the quality requirement that can be measured as the outcome of interactions with a computer system. This requirement can be fulfilled and the end-user will be satisfied if the intended goals are achieved effectively with the use of proper resources.

Molich in 2000 stated that a user-friendly system should fulfill the following five goals, i.e., easy to Learn, easy to remember, efficient to use, satisfactory to use, and easy to understand.

In addition to the different definitions of Usability, there are some standards and quality models and methods that define usability in the form of attributes and sub-attributes such as ISO-9126, ISO-9241-11, ISO-13407, and IEEE std.610.12, etc.

3.6.5 Portability Testing

Portability Testing includes testing a software with the aim to ensure its reusability and that it can be moved from another software as well. Following are the strategies that can be used for portability testing –

- Transferring an installed software from one computer to another.
- Building executable (.exe) to run the software on different platforms.

Portability Testing can be considered as one of the sub-parts of system testing, as this testing type includes overall testing of a software with respect to its usage over different environments. Computer hardware, operating systems, and browsers are the major focus of portability testing. Some of the pre-conditions for portability testing are as follows –

- Software should be designed and coded, keeping in mind the portability requirements.
- Unit testing has been performed on the associated components.
- Integration testing has been performed.
- Test environment has been established.

3.6.6 Integration Testing

Unit Testing focuses on testing of different modules/functions and programs that are small part of entire information system being developed. These modules are expected to work together to achieve objectives of information system. For example, Internet Banking is a system consisting of various functions like saving account management, time deposit management, loan account management, third-party fund transfer, standing instruction, getting statements of accounts etc. While developing programs/functions, each service function is developed separately and tested in Unit Testing. Now it is necessary to test if these modules/functions work together seamlessly and communicate appropriately during execution. Objective is to evaluate the validity of integration of two or more components that pass information to one another. Integration Testing puts together modules that have been unit tested and applies tests defined. There are two approaches for Integration Testing:

1. **Bottom-up Integration:** It is the traditional strategy used to integrate the components of a software system starting from smallest module/function/program. It consists of Unit Testing, followed by Sub-system Testing. Bottom-up testing is easy to implement as at the time of module testing, tested subordinate modules are available. The disadvantage; however, is that testing of major decision / control points is deferred to a later period. For example in above example of Internet Banking it will test communication between different modules using smallest level of module like saving bank account, fund transfer and then statement of accounts to ensure previous transaction reflects in statement, and so on, however it might not ensure the overall control on passing parameters required for session time out or inactive session
2. **Top-down Integration:** This starts with the main routine followed by the stubs being substituted for the modules which are directly subordinate to the main module. Considering above example, the testing will start from opening login screen and then login, then selecting function one by one. An incomplete portion of a program code is put under a function (called stub) to allow the function. Here a stub is considered as Black Box and assumed to perform as expected, which is tested subsequently. Once the main module testing is complete, stubs are

substituted with real modules one by one, and these modules are tested. This process continues till the atomic (smallest) modules are reached. Since decision-making processes are likely to occur in the higher levels of program hierarchy, the top-down strategy emphasizes on major control decision points encountered in the earlier stages of a process and detects any error in these processes. The difficulty arises in the top-down method, because the high-level modules are tested with stubs and not with actual modules.

3.6.7 Regression Testing

It is a testing performed during change management when a function/module/program is changed or added to existing software, in order to ensure that new/changed functions executes properly and integrates with other modules as expected. It is required since new data flow paths are established, new I/O may occur and new control logic is invoked. These changes may cause problems with functions that previously worked flawlessly. In the context of the integration testing, the regression tests ensure that changes or corrections have not introduced new faults. The data used for the regression tests should be the same as the data used in the original test.

3.6.8 System Testing

It is a process in which software and other system elements are tested as a whole. System testing begins either when the software as a whole is operational or when the well-defined subsets of the software's functionality have been implemented. The purpose of system testing is to ensure that the new or modified system functions properly. These test procedures are often performed in a non-production test environment. The types of testing that might be carried out with various other objectives described below:

- **Recovery Testing:** This is the activity of testing 'how well the application is able to recover from crashes, hardware failures and other similar problems. Recovery Testing is the forced failure of the software in a variety of ways to verify that recovery is able to be perform properly, in actual failures
- **Security Testing:** This is the process to determine that an Information System protects data and maintains functionality as intended. The three basic security concepts that required to be covered by Security Testing are – Confidentiality, Integrity and Availability. In addition, the software may further be tested for user management requirements require i.e. Authentication, Authorization, and Non-repudiation and Log maintenance.
- **Stress or Volume Testing:** Stress Testing is a form of testing that is used to determine the stability of a given system or entity based on the requirements and expected data growth. It involves testing beyond normal operational capacity, often to a breaking point,

in order to observe the results. Stress Testing may be performed by testing the application with large quantity of data during peak hours to test its performance.

- **Performance Testing:** Software Performance Testing is performed on various parameters like response time, speed of processing, effectiveness use of a resources (RAM, CPU etc.), network, etc. This testing technique compares the new system's performance with that of similar systems using available industry benchmarks.

3.6.9 Other types of Testing

When any complex application/software is intended for general and wide spread use developers want to make sure that product delivers diverse requirements of general users. Organizations may consider Alpha and Beta testing. For example, Microsoft performs this type of testing on new product before making it available commercially.

Alpha Testing: This is the first stage, often performed by users within the organization by the developers, to improve and ensure the quality/functionalities as per users' satisfaction.

Beta Testing: This is the second stage, generally performed after the deployment of the system. It is performed by the external users, during the real-life execution of the project. It normally involves sending the product outside the development environment for real world exposure and receives feedback for analysis and modifications, if any.

Automated Testing: In software testing, automation of testing is performed using special software (separate from the software being tested) to control the execution of tests and the comparison of actual outcomes with predicted outcomes. Test automation can automate some repetitive but necessary tasks in a formalized testing process already in place, or add additional testing that would be difficult to perform manually.

Integrated Testing: Some organizations rely on integrated test facilities. Test data usually are processed in production-like systems. This confirms the behaviour of the new application or modules in real-life conditions. These conditions include peak volume and other resource-related constraints. In this environment, IS Auditor will perform their tests with a set of fictitious data whereas client representatives use extracts of production data to cover the most possible scenarios as well as some made-up data for scenarios that would not be tested by the production data.

Some organizations use a subset of production data in a test environment, such production data may be altered or scrambled to mask the confidential data. This is often the case where the acceptance testing is done by team members who, under usual circumstances, would not have access to such production data. These tools help in building test cases and also generate test data based on conditions. However, using production data may not help in identifying negative test cases.

Accreditation of Software: Although it is not type of testing, many organizations insist on certification and accreditation of software. Generally, this is done by the software development houses before taking product to market. In case of tailor-made software certification/accreditation should only be performed once the system is implemented and in operation for some time to produce the evidence needed for certification/accreditation processes. This process includes evaluating program documentation and testing effectiveness. The process will result in a final decision for deploying the business application system.

Security Testing: (Application Scans and Penetration Testing) For information security issues, the evaluation process includes reviewing security plans, the risk assessments results along with response decision, and the evaluation of processes to be deployed. The result of security assessment focuses on measuring effectiveness of the security controls. Security testing provides assurance to the business owner.

Security testing of web application for identified external threats (like SQL injection, cross site scripting etc.) is necessary to ensure that the application can sustain an attack by the hacker who is trying to breach the security.

While reviewing testing process IS auditors focus on getting answers to following questions:

1. Whether the test-suite prepared by the testers includes the actual business scenarios?
2. Whether test data used covers all possible aspects of system?
3. Whether CASE tools like 'Test Data Generators' have been used?
4. Whether test results have been documented?
5. Whether tests have been performed in their correct order?
6. Whether modifications needed based on test results have been done?
7. Whether modifications made have been properly authorized and documented?

Testers generally perform Black Box testing (Penetration Test) by trying to simulate attacks on hosted application. This is then followed by performing Grey Box and/or White Box testing that includes code review to identify the issues in coding practices that might introduce the vulnerabilities in the application. These can be avoided by including secure coding practices in coding standard developed by the organizations.

3.7 Final Testing

It is conducted if results of system testing are satisfactory and when the system is just ready for implementation. This testing is performed at two levels:

- At technical level Quality Assurance Testing is performed
- At functional level User Acceptance Testing is performed.

3.7.1 Quality Assurance Testing (QAT)

QAT focuses on conforming to the quality standards of the organization accepted before development. It includes documented specifications, technology employed, use of coding standards, and the application meets the documented technical specifications and deliverables. QAT is performed primarily by the technical (IT) department. The participation of the end user is minimal and on request. QAT does not focus on functionality testing.

3.7.2 User Acceptance Testing (UAT)

It is a user extensive activity and participation of functional user is a primary requirement for UAT. The objective of UAT is to ensure that the system is production-ready and satisfies all accepted (baselined) requirements. UAT is a formal process and may include:

- Definition of test strategies and procedures
- Design of test cases and scenarios
- Execution of the tests
- Utilization of the results to verify system readiness

Acceptance criteria defined along with requirement specifications includes that deliverables must satisfy the predefined needs of the user. A UAT plan must be documented for the final test of the completed system. The tests are written from a user's perspective and should test the system in a manner as close to production as possible. For example, tests may be based around typical predefined, business process scenarios. If new business processes have been developed to accommodate the new or modified system they should also be tested at this point. A key aspect of testing should also include testers seeking to verify that supporting processes integrate into the application in an acceptable fashion. Successful completion would generally enable a project team to hand over a complete integrated package of application and supporting procedures.

- UAT is a stage in SDLC where end users finally accept the developed application system. This is required for all situations of acquiring software i.e. software developed in-house, or by outsourced team or purchased and configured by vendor. A formal sign-off generally marks end of development process.
- UAT should be performed in a secure testing or staging environment where both source and executable code are protected to ensure that unauthorized or last-minute changes are not made to the system unless authorized and the standard change management

process is followed. In the absence of controls, the risk of introducing unauthorized changes/malicious patches/Trojan horse programs is very high.

- Users should develop test cases or use data of live operations of a specified period to confirm whether the processing of data by new application is providing correct results, has required controls and the reports meet the management requirements.

Many organizations expect a report from IS Auditor after tests are completed. The IS Auditor should issue an opinion to management as to whether the system meets documented business requirements, has incorporated appropriate controls, and may be migrated to production. This report should also identify and explain the risk that the organization might be exposed by implementing the system.

3.8 Implementation

Application software developed shall be implemented once it is tested and UAT has been signed off. However, the planning for implementation must start much earlier in SDLC, many times after feasibility study. Planning involves:

- Selecting Implementation Strategies
- Preparing for implementation
 - Deciding on hardware and ordering (if required) in advance so as to be available in time
 - Deciding on site where infrastructure to be made available
- Conversion of data to suit to the requirements of new application.

3.8.1 Implementation Strategies

Considering the nature of business operation appropriate implementation strategy must be decided, much earlier in SDLC. Generally, it is decided once the design is finalized or in case of acquisition once the application is selected. Organization can adopt one of the four strategies, which are described below:

Cut-off or Direct Implementation / Abrupt Change-Over: This is achieved through an abrupt takeover – an all or no approach. With this strategy, the changeover is done in one operation, completely replacing the old system in one go. Fig 3.1 depicts Direct Implementation, which usually takes place on a set date, often after a break in production or a holiday period so that time can be used to get the hardware and software for the new system installed without causing too much disruption.

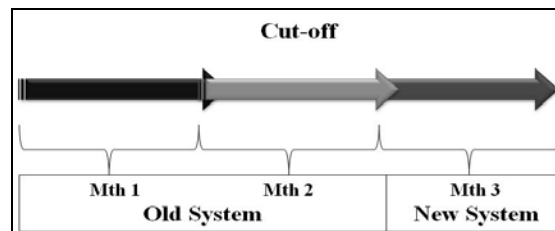


Fig. 3.1: Cut-off or Direct implementation

The challenge in cut-off implementation is that roll-back is most difficult and hence planning must be meticulously done. Also, conversion activity must start well in advance and must be properly planned.

Phased Changeover: With this strategy, implementation can be staged with conversion to the new system taking place gradually. This is done based on business operations. For example, converting one function (e.g. Marketing) on new system, wait for the same to be stabilized and then take another function (Finance/HR/Production etc.) When one phase is successful the next phase is started, eventually leading to the final phase when the new system fully replaces the old one as shown in Fig. 3.2.

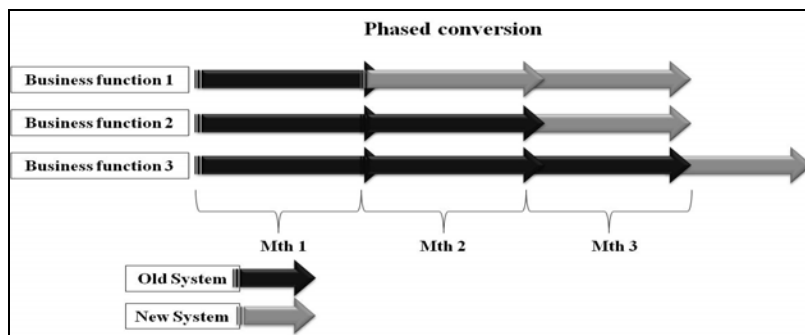


Fig. 3.2: Phased changeover

Phase changeover might require more time to implement however it helps in stabilizing one function before starting another.

Pilot Changeover: With this strategy, the new system replaces the old one in one operational area or with smaller scale. Any errors can be rectified and new system is stabilized in pilot area, this stabilized system is replicated in operational areas throughout the whole system. For example, converting banking operations to centralized systems are done at one branch and stabilized. The same process is replicated across all branches. Fig. 3.3 depicts Pilot Implementation.

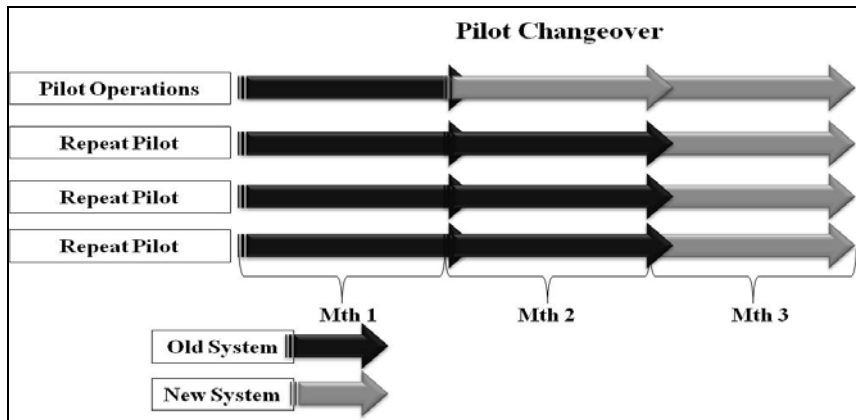


Fig. 3.3: Pilot Changeover

Advantage of pilot implementation is that issues and problems are identified and rectified during pilot run and a stabilized system is implemented thus saving cost and enabling faster implementation.

Parallel Changeover: This is considered the most secure method, time and resource consuming implementation. The new systems is implemented, however the old system also continues to be operational. The output of new system is regularly compared with old system. If results match over period of time and issues observed with new system are taken care of, the old system is discontinued. Fig. 3.4 shows parallel implementation.

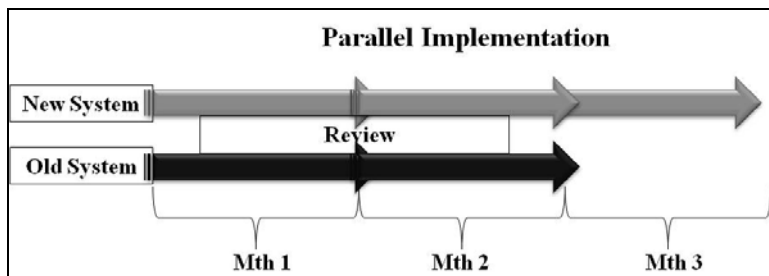


Fig. 3.4: Parallel Changeover

However, it is costly and may not be feasible in large and complex systems, since all transactions must be processed twice. Many times, users may not be conformable in duplicating the work.

3.8.2 Preparing for Implementation

In order to finally deploy or implement the new system in the operating environment, several activities are undertaken. A fully functional as well as documented system is a prerequisite for

implementation to begin. Moreover, many other issues like defect removal, maintenances, reengineering may require to be addressed to assure the desirable quality control of the system in operational environment.

The process of ensuring that the information system is operational and then allowing users to take over its operation for use and evaluation is called System Implementation. Implementation includes all those activities that take place to convert from the old system to the new. The new system may be totally new, replacing an existing manual or automatic system. Some of the generic key activities involved in System Implementation are:

- Site preparation and hardware installation
- Conversion of data to the new system files;
- Training of end users;
- Completion of user documentation;
- System changeover
- Post implementation review and evaluation

Site Preparation and Installation: The hardware required to support the new system is selected prior to the implementation phase. The necessary hardware should be ordered in time to allow for installation and testing of equipment during the implementation phase. An installation checklist should be developed at this time with operating advice from the vendor and system development team. In situations, where people are not experienced in the installation of similar hardware/platform/equipment, adequate time should be planned to allow completion of required activities.

Site Preparation: An appropriate location as required to provide an operating environment for system (temperature, humidity and dust control specifications) has to be prepared in time.

Installation of New Hardware / Software: Site preparation also includes receiving, installing and connecting the hardware and supporting software (like operating systems, middleware etc.). In case the hardware is available, the same needs to be commissioned for the new application as per design requirements.

Equipment Checkout: The equipment must be turned on for testing under normal operating conditions. Though the routine 'diagnostic tests' should be run by the vendor, the in-house implementation team should test the equipment functionalities in actual working conditions.

3.8.3 Conversion

Conversion of data is most important activity while implementing new application system or when there is significant change in technology requiring conversion. The Conversion activity

involves converting data, procedures, documentation from old system to new system. Most important being data conversion.

Data Conversion: The requirement of data conversion depends upon the change. If the new application is replacing manual operation to automated operation it involves:

1. Capturing of data into electronic form
2. Verification of data
3. Uploading into database

In case change is from old system to new system, it involves:

1. Converting electronic data from old format to new format
2. Verification
3. Uploading into new database.

Since data conversion is a type of input, controls on conversions are essential to ensure integrity of data. These controls generally include:

1. **Completeness Check:** Using number of records, control totals, batch totals, hash totals. For example, verifying number of employee's record, checking trial balance before and after conversion etc.
2. **Accuracy Check:** Manual verification or key verification (manual to electronic conversion)

Unauthorized changes during conversion are one of the sources of frauds.

Procedure Conversion: Changes in application systems may require changes in operating procedures and associated controls. Operating procedures should be carefully completed with sufficient documentation pertaining to operations on how to use the new system. It applies to both computer-operations and functional area operations. Before conversion activities can start, conversion procedures must be defined and personnel involved must be trained to cover input, data files, methods, procedures, output, and internal control.

For example, during manual operation in banking every transaction is verified before being posted to account and then the effect of transaction is reflected in general ledger. However in electronic banking system transactions are flagged with type of transaction and posted to general ledger. Hence verification of transaction is most essential in new system.

System Conversion: After on-line and off-line files have been converted and the reliability of the new system has been confirmed for a functional area, daily processing can be shifted from the existing information system to the new one. All transactions initiated after this time are processed on the new system. System development team members should be present to

assist and to answer any questions that might develop. Consideration should be given to operating the old system for some more time to permit checking, matching and balancing the total results of both systems.

Scheduling Personnel and Equipment: Scheduling data processing operations of a new information system for the first time is a difficult task for the system manager. As users become familiar with the new system, the job becomes easier to perform and becomes part of the routine work. Schedules should be set up by the system manager in conjunction with departmental managers of operational units serviced by the equipment. The master schedule for next period/month should provide sufficient computer time to handle all required processing.

3.9 Change Management Process

Application maintenance refers to the process of managing changes in the application and IT triggered or prompted due to changes in processes, regulatory compliances, and strategic changes in business, technology changes and so on. Changes also arise due to issues, problems, incidents faced. In order to handle changes organization should have a defined process. This process generally includes:

1. **Raising Change Request:** Formal process for requesting change. Anyone can raise the Change Request with reason for the change, a cost justification analysis, if possible and the expected benefits of the change. An automated process for raising Change Requests helps in capturing all associated changes and maintaining record of change requests.
2. **Defining Requirements:** Defining details of changes required, like functional changes, appearance changes, processing changes. (e.g. change in tax structure may require processing changes, if tax slabs are displayed then appearance changes and so on)
3. **Analysing Requirements:** Getting answers to the questions such as: why change is required, when it should be effective, who needs it, where the changes are required, what programs/modules/function is affected, how the changes will be carried out and so on.
4. **Impact Analysis:** What will be impact of changes on processes and other related programs that interface with application that need to be changed or how changes in technology shall affect the processing.
5. **Approval of change:** Changes must be approved by the asset owners, i.e. application owners in case of application change and other stakeholders that might be impacted. Sometimes it is difficult to decide who has appropriate authority to approve change due to impact on multiple processes. To overcome such situations organizations forms a Change Approval Board or Committee (CAB) consisting of representatives from multiple business functions.

6. **Prioritizing the Change Requests:** This is required to resolve the conflict due to multiple Change Requests from different users.

7. **Carrying out Changes:** System Analyst shall review the changes and decide appropriate resources to carry out changes. Records of all program changes should be maintained. Library management software may help in automating this process and also maintaining audit trail. The maintenance information usually consists of the programmer ID, time and date of change, project or request number associated with the change, and before and after images of the lines of code that were changed. This also helps in preventing and/or detecting unauthorized changes.

8. **System Document Maintenance:** All relevant System Documentation updating sometimes is neglected area during change management. It is essential to ensure the effective utilization and future maintenance of a system, Documentation requiring revision may consist of program and/or system flowcharts, program narratives, data dictionaries, entity relationship models, data flow diagrams (DFDs), operator run books and end-user procedural manuals. In case of infrastructure changes network diagrams, data centre block diagrams, electrical and facility diagrams etc. are likely to undergo changes.

9. **Testing the Changes:** Changes will be tested as per testing process (Please refer subsection on testing). However, for testing changes, following points must be considered:

- Existing functionalities are not affected by the change
- System performance is as expected
- Security vulnerabilities are not introduced

This also includes conducting user acceptance testing and formal sign-off from users/owners. (E-mail, electronic approval in automated system, document etc.)

10. **Releasing Changes:** Changes shall be released to production once approved by stakeholders (UAT). Ensure fall-back procedures in place in case operations are affected due to change. Automation of this process shall help management in restricting one person requiring access to production, test and development environment.

11. **Review:** Post implementation/release review may be conducted.

12. **Record Maintenance:** Change Requests should be maintained in a format that will ensure that all changes associated with primary change requests are considered. This allows the management to easily track the changes to change requests. The process must be formal and maintain record of all approvals and rejections.

For acquired systems vendor may distribute periodic updates, patches or new version of the software. User and systems management should review such changes for impact and appropriateness before implementing.

While reviewing change management IS Auditor should ensure that:

- Access to source program is restricted.
- Change Requests are approved and record is maintained to trace and track the changes.
- Impact assessment is performed before approving changes.
- The Change Request should be documented in standard format covering at the minimum:
 - Change specifications, benefit analysis developed and a target date.
 - Change form has been reviewed and impact assessment is recorded.
 - Change Request has been approved formally
- Verify records of changes for sample changes made and trace end-to-end (from request till closure) confirm that the changes are authorized, approved, and moved to production after UAT.

3.9.1 Emergency Changes

In exceptional situations there may be need to make changes to production to resolve issues in time. This requirement can arise due to one or more reasons like:

- Events/incidents
- Short notice requirement changes (due to external incidents/events: terrorist attacks, natural disasters, etc.)
- Infrastructure failure
- Production issues due to unexpected data conditions

Procedures should focus to ensure that emergency changes can be performed without compromising the integrity of the system. Organization should have a process for carrying out emergency changes. The process may consist of following steps:

1. Identify need for emergency change (process issue, incident/event etc.)
2. Determine activities involved. Generally, it may involve providing all accesses to one person. A special user ID may be created with higher privileges for this purpose and all activities are logged and reviewed.
3. A post-facto change management process must be followed, so as to ensure consistency in documents, source-code library, network diagram etc. as applicable.

The IS Auditor has to ensure that emergency changes are handled in a controlled manner.

3.9.2 Implementing Changes into Production

Changes are implemented into production environment once they are approved by the user management (UAT sign-off). The best practices suggest that this implementation should be done by independent team not involved in development or testing of changes. In case of client-server applications or distributed systems, such as point-of-sale systems, the process should be properly documented and implemented over a period of time to ensure:

- Conversion of data
- Training of users
- Support process for changes
- Rollback plan
- All points are updated

3.9.3 Segregation of Duties

Uncontrolled change management has risk associated with unauthorized changes. An unauthorized change occurs due to various reasons:

- Developer has access to production libraries containing programs and data including object code.
- User has not approved change or not aware of the change
- A change procedure has not been formally established.
- A change was updated into production without user approval.
- The change has not been reviewed or tested.
- Developer inserted extra logic for personal benefit (i.e., committed fraud).
- In case of vendor software, changes received were not tested.

In order to control unauthorized changes segregation of duties has to be implemented at organization level. The typical segregation includes following controls at the minimum:

- Development, Test and Production environments to be physically separated.
- Developers' team, Testing team and Production user should not have access to other areas. I.e. developers should not have access to test and production and so on.
- Source code must be maintained by librarian. At least control must be in place to prevent or detect insertion of unauthorized code.

- A separate change control team or release team should be appointed to move source/object code from development to test and from test to production.

It may not be possible for some organizations to implement strict segregation of duties, in such situation appropriate compensating controls should be present to prevent or detect and correct unauthorized changes. Some such situations may arise due to:

1. The developer is also the operator due to small IT department. In this case user management required to ensure proper authorization and monitoring of changes and upgrades made by the programmer.
2. Emergency changes to resolve the issues in production.
3. In case separate release team is not possible, compensating control by enabling user ID of user who moves changes from development to test and/or test to production only after approval, and monitoring activities may work as compensating control.
4. Developers should not have written, modify or delete access to production data. Depending on the type of information in production, programmers may not have read-only access to personally identifiable information.

3.9.4 Configuration Management

Configuration Management refers to automated processes that organizations install to maintain all information assets and work-flows required to maintain them. The backend of such system is a data base called a Configuration Management Data Base (CMDB); hence sometimes the system is also referred to as CMDB.

Configuration Management system helps in maintaining information about system as a collection of Configuration Items (CI). A CI can be a module/function/program or database instance of IT asset associated with a system and referenced with an ID. Workflows around the CMDB consist of workflows for Change Management, configuration management etc. Change Management requests must be formally documented and approved by a change control group within CMDB. CMDB then manages the change process via checkpoints, reviews and sign-off procedures that generates audit trails.

Configuration Management sometimes may provide procedures throughout the software life cycle (from requirements analysis to maintenance) to identify, define and baseline software items in the system and thus provide a basis for Problem Management, Change Management and Release Management. However, though it sounds easy, proper implementation of CMDB is a necessary requirement (which must follow SDLC process for acquired Software).

Software Configuration Management requires following tasks to be performed:

1. Develop Configuration Management Plan.

2. Baseline Application and Associated Assets.
3. Analyze results of Configuration Control.
4. Develop Monitoring of Configuration Status.
5. Develop Release Procedures.
6. Define and implement Configuration Control activities (such as identification and recording of Change Requests.)
7. Update the Configuration Status Accounting Database.

A Configuration Management Tool supports change and release management by supporting following activities:

1. Identification of items affected by a proposed change
2. Help in impact assessment by providing information
3. Recording configuration items affected by changes
4. Implementation of changes as per authorization
5. Registering of configuration item changes when authorized changes and releases are implemented
6. Recording of original configuration to enable rollback if an implemented change fails
7. Preparing a release to avoid human errors and resource costs

3.10 Summary

Testing is a process that focuses on correctness, completeness and quality of developed computer software. Although the testing phase comes much later in the life cycle, planning for testing starts with the commencement of System Development Life Cycle i.e. during requirement gathering phase. Testing should systematically uncover different classes of errors in a minimum amount of time with a minimum amount of efforts. The data collected through testing can also provide an indication of the software's reliability and quality. However, testing cannot show the absence of defect, it can only show that software defects are present.

3.11 Questions

1. Which of the following is main reason to perform User Acceptance Test (UAT)?
 - A. To train and educate users on features of new solution.
 - B. To confirm from users that solution meets requirements.
 - C. To complete formality of sign-off to mark end of project.
 - D. To finalize the implementation plan for new IT solution.

2. An organization has developed a web-based application for the use of internal users to be hosted on intranet. Before finalizing and making it live it was decided to make it available to users for providing feedback. This is an example of:
 - A. Internal Audit
 - B. Alfa Testing
 - C. Beta Testing
 - D. User Training
3. A major concern associated with using sanitized old production data for testing new application is that:
 - A. User may not provide sign off.
 - B. Production data may be leaked.
 - C. Integration testing cannot be performed.
 - D. All conditions cannot be tested.
4. A tester is executing a test to evaluate that it complies with the user requirement that a certain field be populated by using a dropdown box containing a list of values. Tester is performing _____
 - A. White-Box Testing
 - B. Black-Box Testing
 - C. Load Testing
 - D. Regression Testing
5. What is the order in which test levels are performed?
 - A. Unit, Integration, System, Acceptance
 - B. Unit, System, Integration, Acceptance
 - C. Unit, Integration, Acceptance, System
 - D. It depends on nature of a project
6. Which testing is concerned with behavior of whole product as per specified requirements?
 - A. Acceptance Testing
 - B. Component Testing
 - C. System Testing
 - D. Integration Testing

7. Verifying that whether software components are functioning correctly and identifying the defects in them is objective of which level of testing?
 - A. Integration Testing
 - B. Acceptance Testing
 - C. Unit Testing
 - D. System Testing
8. Which technique is applied for usability testing?
 - A. White Box
 - B. Black Box
 - C. Grey Box
 - D. Combination of all
9. If a company decides to migrate from Windows XP to Windows 7, which type of testing is done to ensure whether your software works on new platform?
 - A. Interoperability Testing
 - B. Portability Testing
 - C. Usability Testing
 - D. Performance Testing
10. Boundary value analysis belongs to?
 - A. White Box Testing
 - B. Black Box testing
 - C. White Box & Black Box testing
 - D. None of the above

3.12 Answers and Explanations

1. B is the correct answer. UAT is mainly conducted to confirm from the users and application owners that application meets their requirements. Option C is a formality to be completed only if requirements are met. Training and implementation planning are different activities which are not dependent on UAT.
2. C is the correct answer. Beta testing is making product available to users for feedback before launching. Option A Internal Audits seek to identify any shortcomings in a company's internal controls. Option B Alpha Testing is performed by the developers to

identify bugs before releasing the product to real or intended users. Option D User Training helps successful system implementation.

3. D is the correct answer. Sanitized data generally may not cover all paths the data can take and hence system cannot be tested for all possible cases. Option B leakage of production data is not a major concern since data is sanitized. Options A and C are not concerns.
4. B is the correct answer. Black Box testing focuses on the inputs and outputs without knowing their internal code implementation. Option A White Box testing evaluates the code and the internal structure of a program. Option C Load Testing is performed to determine a system's behaviour under both normal and at peak conditions. Option D Regression Testing is defined as a type of software testing to confirm that a recent program or code change has not adversely affected existing features.
5. D is the correct answer. Test levels can be combined or reorganized depending upon nature of a project or system architecture. Unit testing refers to test a function, individual program or even a procedure. Integration Testing allows individuals to find interface defects between the modules/functions. System Testing is the first level in which the complete application is tested as a whole. Acceptance Testing (or User Acceptance Testing) determines whether the system is ready for release.
6. C is the correct answer. System Testing is based on Functional Requirement Specification (FRS), which tells about general behavior of a system. Acceptance testing (or User Acceptance Testing) determines whether the system is ready for release. Component Testing, also known as Unit, Module or Program Testing, is defined as a software testing type, in which the testing is performed on each individual component separately without integrating with other components. Integration testing allows individuals to find interface defects between the modules/functions.
7. C is the correct answer. Separately testable components are tested in Unit Testing or Component Testing. A Unit Testing tends to test a function, individual program or even a procedure. Option B Acceptance Testing (or User Acceptance Testing) determines whether the system is ready for release. Option A Integration Testing allows individuals to find interface defects between the modules/functions. Option D System Testing is the first level in which the complete application is tested as a whole.
8. B is the correct answer. Usability Testing is mostly done by users. They are not familiar with internal structure of the system and hence Black Box technique is correct answer. Option A White Box testing evaluates the code and the internal structure of a program. Option C Grey Box testing is a process for debugging software applications by making an input through the front-end, and verifying the data on the back-end. Option D does not exist.

9. B is the correct answer. Portability Testing shows the ease with which a computer software component or application can be moved from one environment to another, e.g. moving of any application from Windows XP to Windows 7. Option A Interoperability testing checks whether software can inter-operate with other software component, software or systems. Option C Usability Testing, is a non-functional testing technique that is a measure of how easily the system can be used by end users. Option D Performance Testing is the process of determining the speed, responsiveness and stability of a computer, network, software program or device under a workload.
10. B is the correct answer. Boundary Value Analysis is based on testing at the boundaries between partitions and checks the output with expected output. Option A White Box testing evaluates the code and the internal structure of a program. Option C also known as Grey Box testing is a process for debugging software applications by making an input through the front-end, and verifying the data on the back-end. Option D is not applicable.

Chapter 4

Application Controls

Learning Objectives

After studying this chapter, you will be able to have basic understanding of Application Controls and types of Application Controls. You will also learn the mapping of Application Controls to business processes, and the role of auditors.

4.1 Introduction

Most Application Control solutions also allow for visibility into applications, users, and content. This is helpful for understanding the data of the enterprise and controls, its storage locations, which users have access to it, the access points, and the data transmission process. These steps are required for data discovery and classification for risk management and regulatory compliance. Application Control supports these processes and allows organizations to keep their finger on the pulse of what is happening within their network.

Application Control gives companies and organizations knowledge about key areas regarding applications, web traffic, threats, and data patterns. Users can also benefit from Application Control by gaining a better understanding of applications or threats, applications' key features and behavioral characteristics, details on who uses an application, and details on those affected by a threat. Organizations also gain knowledge about traffic source and destination, security rules, and zones to get a complete picture of application usage patterns, which in turn allows them to make more informed decisions on how to secure applications and identify risky behavior. While they are making those decisions, the Application Control solution is automatically protecting the network with whitelisting and blocking capabilities.

4.2 What is Application Control?

Application Control is a security practice that blocks or restricts unauthorized applications from executing in ways that may put data at risk. The control functions vary based on the business purpose of the specific application, but the main objective is to help ensure the privacy and security of data used by and transmitted between applications.

Application Control includes:

- Logical access controls (i.e., those that limit access to application functionality)
- Data entry/field validations (e.g., validation of entered credit card numbers)
- Business rules

- Work flow rules (e.g., routing and sign-off of purchase requests)
- Field entries being enforced based on predefined values (e.g., pricing information)
- Work steps being enforced based on predefined status transitions (e.g., open > reviewed > closed)
- Reconciliations
- Review and follow-up of application-generated exception reports
- Automated activity logs
- Automated calculations
- Management and audit trails

Simply put, Application Controls ensure proper coverage and the Confidentiality, Integrity, and Availability of the application and its associated data. With the proper Application Controls, businesses and organizations greatly reduce the risks and threats associated with application usage because applications are prevented from executing if they put the network or sensitive data at risk.

4.2.1 Features and Benefits of Application Control

Companies have grown increasingly dependent upon applications in day-to-day business operations. With web-based, cloud-based, and third-party applications at the core of today's business processes, companies are faced with the challenge of monitoring and controlling data security threats while operating efficiently and productively. Most Application Control solutions include whitelisting and blacklisting capabilities to show organizations which applications to trust and allow to execute and which to stop. With Application Control, companies of all sizes can eliminate the risks posed by malicious, illegal, and unauthorized software and network access.

Key features and benefits of Application Control:

- Identify and control which applications are in your IT environment and which to add to the IT environment
- Automatically identify trusted software that has authorization to run
- Prevent all other, unauthorized applications from executing – they may be malicious, untrusted, or simply unwanted
- Eliminate unknown and unwanted applications in your network to reduce IT complexity and application risk
- Reduce the risks and costs associated with malware

- Improve your overall network stability
- Identify all applications running within the endpoint environment
- Protect against exploits of unpatched OS and third-party application vulnerabilities

4.3 Types of Application Controls

Application Controls are controls over input, processing and output functions. Application Control ensures that:

- Only complete, accurate and valid data are entered and updated in a information system
- Processing accomplishes the intended task
- Processing results meet expectations and
- Integrity of data is maintained

4.3.1 Input Controls

Input Control procedures must ensure that every transaction to be processed is entered, processed and recorded accurately and completely. These controls must ensure that only valid and authorized information is input and that these transactions are processed once only. In an integrated systems environment, output generated by one system is the input for another system.

Input Authorization verifies that all transactions have been authorized and approved by management. Types of authorization include:

- Signature on batch forms and source documents
- Online access controls
- Unique passwords
- Terminal or client workstation identification
- Source documents

Batch Controls group input transactions to provide control totals. The batch control can be based on total monetary amount, total items, total documents or hash totals.

Input processing require that controls be identified such that only correct data are accepted into the system and input errors are recognized and corrected.

Input Error Handling can be processed by:

- Rejecting only transactions with errors

- Rejecting the whole batch of transactions
- Holding the batch in suspense
- Accepting the batch and flagging error transactions

Ideally all source documents should be appropriately controlled.

4.3.2 Processing Controls

Processing Controls ensure the reliability of application program processing. IS Auditor need to understand the procedures and controls that can be evaluated.

Data Validation and Editing Procedures should be established to ensure that input data are validated and edited as close to the time and point of origination as possible. There should be system of logging in case any override happens and logs should be reviewed.

Processing Controls ensure the completeness and accuracy of accumulated data. Some of the processing control techniques are:

- Manual recalculation
- Editing
- Run-to-run totals
- Programmed controls
- Reasonable verification of calculated amounts
- Limit check on amounts
- Reconciliation of file totals
- Exception reports

Data File Procedures ensure that only authorized processing occurs to stored data. Data file controls are:

- Before and after image processing
- Maintenance of error reporting and handling
- Source documentation
- Internal and external labeling
- Version usage
- Data file security
- One-for-one checking

- Pre-recorded input
- Transaction logs
- File updating and maintenance authorization
- Parity checking

The control over data files or database tables are of four categories:

- System control parameters
- Standing data
- Master data/ balance data
- Transaction files

4.3.3 Output Controls

Output Controls provide assurance that the data delivered to users will be presented, formatted and delivered in a consistent and secure manner. These include:

- Logging and storage of negotiable, sensitive and critical forms in a secure place
- Control over computer generated negotiable instruments, forms and signatures
- Report accuracy, completeness and timeliness
- Report generated from the system
- Report distribution
- Balancing and reconciling
- Output error handling
- Output report retention
- Verification of receipt of reports

4.3.4 Business Process Control Assurance

In an integrated application environment, controls are embedded and designed into the application that supports the processes. Business Process Control Assurance evaluates controls at process and activity level and may be a combination of management, programmed and manual controls. In general Business Process Control Assurance considers:

- Process and data flow mapping
- Process controls
- Assessing business risks within the process

- Benchmarking with best practices
- Roles and responsibilities
- Activities and tasks
- Data restrictions

4.4 Application Control Objectives

Application Controls are intended to provide reasonable assurance that management's objectives relative to a given application have been achieved. Management's objectives are typically articulated through the definition of specific functional requirements for the solution, the definition of business rules for information processing and the definition of supporting manual procedures. Examples include:

- **Completeness**—The application processes all transactions and the resulting information is complete.
- **Accuracy**—All transactions are processed accurately and as intended and the resulting information is accurate.
- Application Controls can be viewed as those policies, procedures and activities designed to provide reasonable assurance that objectives relevant to a given automated solution are achieved.
- **Validity**—Only valid transactions are processed and the resulting information is valid.
- **Authorization**—Only appropriately authorized transactions have been processed.
- **Segregation of Duties**—The application provides for and supports appropriate segregation of duties and responsibilities as defined by management.

To satisfy business objectives, information needs to conform to certain control criteria:

- **Effectiveness**—Deals with information being relevant and pertinent to the process as well as being delivered in a timely, correct, consistent and usable manner
- **Efficiency**—Concerns the provision of information through the optimal (most productive and economical) use of resources
- **Confidentiality**—Concerns the protection of sensitive information from unauthorized disclosure.
- **Integrity**—Relates to the accuracy and completeness of information as well as to its validity in accordance with business values and expectations.
- **Availability**—Relates to information being available when required by the process now and in the future. It also concerns the safeguarding of necessary resources and associated capabilities.

- **Compliance**—Deals with complying with the laws, regulations and contractual arrangements to which the process is subject, i.e., externally imposed business criteria as well as internal policies.
- **Reliability**—Relates to the provision of appropriate information for management to operate the entity and exercise its fiduciary and governance responsibilities

4.5 Designs and Implementation of Application Controls

Enterprises regularly consider business and functional requirements as part of application design, but mostly do not explicitly consider 'control' requirements. This can create implementation and operational challenges if necessary, controls are not built into the solution from the start and a 'retrofit' of control activities post-implementation is required. In addition to the costs associated with fixing integrity problems, retrofitting controls post-implementation can be very costly. Management should ensure that control requirements are appropriately identified, based on the business risks, and included in functional requirements.

Management can optimize the efficiency and effectiveness of its control design through a balance of various attributes, types and nature of control activities. For example:

- Should a given control activity be a manual activity, automated or some combination of both—a hybrid control? If automated, should the control be designed to be 'configurable' to facilitate changes to business rules over time?
- Is it more cost-effective/efficient to design the control activity to prevent errors from occurring or to design a procedure that would detect any error situations should they arise?
- Is the frequency of the control, the proximity of the control activity to the risk event and the role of the individual performing the activity going to be sufficient to reduce the risk of error conditions to an acceptable level?
- Will the benefits to be realized from reducing a risk outweigh the cost of building, testing and performing the added control activity?

Because testing validates whether or not the designed control activities operate as they were intended, it is essential that the systems accreditation activities include testing of these Application Control activities. Having a clearly documented trail of testing automated Application Controls, the automated components of hybrid and configurable controls may also provide the necessary evidence to demonstrate the effective operation of these controls. Having a clearly documented trail of testing/ validation of manual controls, the manual activities associated with hybrid controls can reinforce their viability and user understanding of the activities.

In approving the design and implementation of Application Controls, management needs to consider the relative efficiency and effectiveness associated with various control design choices and be satisfied that the controls designed are cost-effective and achieve the control objectives, and the relevant information criteria are satisfied.

Assessing risks, identifying relevant control objectives and determining the sufficiency of design of Application Controls are as relevant to existing applications as they are to new applications being acquired/developed and implemented.

Automated Application Controls should be used wherever possible to provide a more cost-effective and sustainable system of internal controls, but they require effective IT general controls.

Responsibility for design and implementation of Application Controls is shared. Business management is accountable for ensuring that Application Control requirements have been appropriately designed and implemented to meet the business objectives. IT management is accountable for developing Application Controls in accordance with business requirements.

4.6 Application Controls and the System Development Life Cycle

A number of SDLC models exist to develop or acquire the application systems to meet the needs of enterprises. The 'Waterfall' SDLC approach is perhaps the best known of these models and is based on systematic, sequential phases of application development (or third-party purchase) in which the output of each stage becomes the input for the next. Iterative development approaches such as Agile are also becoming popular. Agile includes multiple repetitions (or iterations) in small, workable pieces of functionality. Each iteration passes through the development cycle, including planning, requirements analysis, design, coding and testing, with a focus on delivering measurable business value early, continually improving it and adding functionality throughout the life cycle of the project. Regardless of the SDLC approach that enterprises follow, integrating the design, development and implementation of Application Controls is an important step to ensure that the information criteria and management's control objectives are met from the outset of system implementation. Defining Application Controls should be a discrete step in each SDLC process, along with steps associated with defining other business functionality requirements.

Some enterprises use enterprise data modeling to generate an integrated view of the data produced and consumed across the enterprise. An enterprise data model represents a single integrated definition of data, independent of 'how' the data are collected, stored, processed or accessed. As part of defining its data, an enterprise can include a complete range of business requirements for those data and associated information systems, including the CobiT 2019 information quality criteria.

4.7 Business Processes and Application Controls

Business Process controls are activities designed to achieve the broad range of management objectives for the process as a whole. Application Controls, on the other hand, are the sub-set of Business Process controls that relate specifically to the applications and related information used to enable those business processes. Figure 4.1 illustrates the relationship between Application Controls and Business Process controls.

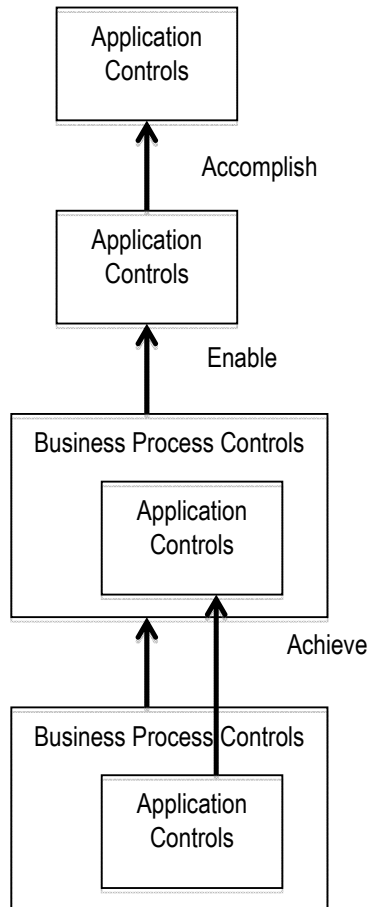


Figure 4.1 Business Goals and Objectives

4.7.1 Business Risks and Information Processing

There are a number of risks associated with any business process and complex processing of business information will introduce further risks. While automated solutions can be much more reliable than manual procedures, this will be the case only if the key risks within the automated solutions have been identified and appropriate controls have been implemented. While not intended to be comprehensive, examples of some key information-related risks and information processing-related risks include:

Incomplete and/or inaccurate information processing—This risk relates to errors that may be made during the collection, input or processing of information.

Invalid or unauthorized transactions being processed—While the previous risk relates to errors that may be made relative to processing legitimate business transactions, this risk relates to the risk of erroneous or illegitimate transactions being processed.

Unauthorized changes to standing data—This is the risk of unauthorized changes to information subsequent to processing by the system.

Bypasses, overrides, manual entries that circumvent controls—This is the risk of misuse of bypasses, overrides or manual entries to avoid automated Application Controls (these functions are inherent in most, if not all, application systems).

Inefficiencies—This risk relates to incurring unnecessary cost or delays during the collection, input, processing, output or transfer of information.

Loss of confidentiality—This risk relates to the inadvertent or intentional disclosure of information that has been identified by management to be sensitive or confidential (such as for business or regulatory compliance reasons).

Unavailability of information—Information is not available when required, causing unnecessary processing delays and inability to make appropriate decisions.

Lack of integrity—This risk relates to lack of reliability of data processed.

4.8 Application Controls Assurance

What is Assurance?

Formal standards such as the International Auditing and Assurance Standards Board's (IAASB's) International Framework for Assurance Engagements (IAASB Assurance Framework) may be referenced for concepts and guidance for assurance. However, these standards are developed and presented from the perspective of an independent auditor providing assurance to third parties. In this publication, 'Assurance' is used in a broader context than 'Audit' and covers evaluation activities not governed by internal and/or external auditing standards.

Common examples of Assurance

Common examples of situations involving the provision of Assurance include:

- **Financial Statement Audit Opinion**—The opinion of the independent auditors (Assurance provider) to the board of directors and shareholders (interested parties) that the enterprise's financial statements (subject matter) are fairly stated (conclusion) in accordance with Generally Accepted Accounting Principles (criteria)
- **Internal Audit Report on review of a given business process**—Report by the Internal Auditor (Assurance provider) to management and the board of directors (interested parties) that the risks within the given business process (subject matter) are being appropriately mitigated (conclusion) based on the COSO ERM framework (criteria)
- **ISO 27001 Accreditation**—An accreditation, often for public display (interested parties), as a result of an examination conducted by an authorized accreditation enterprise (Assurance provider) that the enterprise's Information Security Management System (subject matter) complies with the criteria established by ISO 27001 (criteria)
- **Service Auditor Reports**—The audit and corresponding opinion provided by an independent service auditor (Assurance provider) that the internal control activities of the service enterprise (subject matter) have been appropriately designed and operate effectively (conclusion) to achieve control objectives of interest to the user enterprises and their auditors (interested parties)
- **Management Assertion on Internal Controls as required by Sarbanes-Oxley Section 404**—The assertion by management (Assurance provider) to the shareholders and capital markets (interested parties) that internal controls over financial reporting have been appropriately designed and are operating effectively (conclusion), in accordance with an internal control framework such as COSO (criteria)
- **CIO 'Sub-Certification to the Chief Financial Officer (CFO)/CEO as to the reliability of IT general controls**—The 'Certification' by the CIO (assurance provider) that the IT general controls within his/her span of control and relevant to financial reporting (subject matter) have been appropriately designed (conclusion) in accordance with CobIT (criteria) and are operating effectively (conclusion)

4.8.1 Assurance over Application Controls

Application Controls relate to the transactions and master file, or standing data pertaining to each automated application system, and are specific to each application. They ensure the accuracy, integrity, reliability and confidentiality of the information and the validity of the entries made in the transactions and standing data resulting from both manual and automated processing.

The objectives relevant for Application Controls generally involve ensuring that:

- Data prepared for entry are authorized, complete, valid and reliable.
- Data are converted to an automated form and entered into the application accurately, completely and on time.
- Data are processed by the application accurately, completely and on time, and in accordance with established requirements.
- Data are protected throughout processing to maintain integrity and validity.
- Output is protected from unauthorized modification or damage and distributed in accordance with prescribed policies.

Providing assurance over Application Controls typically involves an assurance provider (the process/application owner, internal auditor, external auditor, etc.) following a process for gathering sufficient evidence that the Application Controls (subject matter) are appropriately designed and are operating effectively (conclusion) relative to established criteria (such as COBIT Application Control objectives).

Materiality

Materiality needs to be considered in determining whether a given set of Application Controls is sufficient to satisfy the control objectives and criteria. The assessment of what is material is a matter of professional judgment and includes consideration of the potential effect on the enterprise's ability to meet its business objectives in the event of errors, omissions, irregularities and illegal acts that may arise as a result of control weaknesses. Materiality can be used as a:

- Factor in determining the amount of evidence necessary to support the assurance provider's conclusion
- Measure of the significance of a finding relative to the subject matter

When conducting or supporting financial statement audits, assurance providers ordinarily measure materiality in monetary terms since what they are auditing is also measured and reported in monetary terms. Application Control assurance providers may be required to provide assurance on non-financial systems (e.g., Air Traffic Control system) or records (e.g., Healthcare Diagnostic Codes) and, therefore, alternative measures are required. With respect to a specific control objective, a material control is a control or group of controls without which control procedures do not provide reasonable assurance that the control objective will be met. ISACA IS Auditing Guideline G6 Materiality Concepts for Auditing Information Systems specifies that where the assurance objective relates to systems or operations processing financial transactions, the value of the assets controlled by the system(s) and the value of transactions processed per day/week/month/year should be considered in assessing materiality.

For systems and controls not affecting financial transactions, the following are examples of measures that could be considered to assess materiality:

- Criticality of the business processes supported by the system or operation
- Cost of the system or operation (e.g., hardware, software, staff, third-party services, overhead costs and/or a combination of these)
- Potential cost of errors (possibly in terms of reputational risk, loss of client/consumer trust, lost sales, warranty claims, irrecoverable development costs, cost of publicity required for warnings, rectification costs, health and safety costs, unnecessarily high costs of production, high wastage)
- Number of accesses/transactions/inquiries processed per period
- Nature, timing and extent of reports prepared and files maintained
- Nature and quantities of materials handled (e.g., where inventory movements are recorded without values)
- SLA requirements and cost of potential penalties
- Penalties for failure to comply with legal and contractual requirements
- Loss of end-user productivity
- Degradation of end-user efficiencies

Detection Risk is the risk that an incorrect conclusion is reached by the Assurance provider regarding the presence (or absence) of material misstatement of the subject matter. In the context of Application Controls, the risk of an incorrect conclusion could, for example, be the risk of concluding that the Application Controls operated effectively when, in reality, they did not. Detection risk is a function of the risk of material error or control failure and the risk that the Assurance provider will not detect associated errors or control failures. The risk of material error has two components:

- **Inherent Risk**—The susceptibility of the subject matter (such as an assertion by the responsible party) to a misstatement that could be material
- **Control Risk**—The risk that a material misstatement could occur in an assertion and not be prevented, detected or corrected on a timely basis by the entity's internal controls. When planning an assurance activity, it is important to consider the inherent risk associated with the subject matter to determine the nature and extent of procedures and to design those procedures to reduce detection risk to an acceptable level.

4.9 Summary

Application Control may consist of edit tests, totals, reconciliations and identification and reporting of incorrect, missing or exception data. Automated controls should be coupled with manual procedures to ensure proper investigation of exceptions. These controls help ensure

data accuracy, completeness, validity, verifiability, and consistency, thus achieving data integrity and data reliability. Implementation of Application Controls helps ensure system integrity, that applicable system functions operate as intended, and that information contained by the system is relevant, reliable, secure and available when needed.

4.10 Questions

1. A company's labour distribution report requires extensive corrections each month because of labour hours charged to inactive jobs. Which of the following data processing input controls appears to be missing?
 - A. Completeness Test
 - B. Valid Code Check
 - C. Limit Test
 - D. Control Total
2. A customer inadvertently orders part number 1234-8 instead of 1243-8. Which of the following controls would detect this error during processing?
 - A. Hash Total
 - B. Check Digit
 - C. Limit Check
 - D. Financial Batch Total
3. Which of the following are not Application Controls?
 - A. Numerical Sequence Check
 - B. Access Security
 - C. Manual follow-up of Exception Reports
 - D. Chart of Accounts
4. Which of the following ensures completeness and accuracy of accumulated data?
 - A. Processing Control Procedures
 - B. Data File Control Procedures
 - C. Output Controls
 - D. Application Controls

5. An integrated test facility is considered a useful audit tool because it:
- A. Is a cost-efficient approach to auditing Application Controls.
 - B. Enables the financial and IS Auditors to integrate their audit tests.
 - C. Compares processing output with independently calculated data.
 - D. Provides the IS Auditor with a tool to analyze a large range of information.

4.11 Answers and Explanations

1. B is the correct answer. It may check the validity and concurrency of the job code. Option A is used for checking the integrity of the data. Option C is used for keeping input up to a certain limit and option D is a figure calculated by the system, adding the values in one of the fields in a segment. This field is called the control totals key figure field.
2. B is the correct answer. It checks the transposition of the digits. Option A is used for checking the integrity of the data. Option C is used for keeping input up to a certain limit and option D is used to check the integrity of all records.
3. B is the correct answer. Access Security is not part of application domain. However options A, C and D are part of the Application Controls.
4. A is the correct answer. Processing controls ensure the completeness and accuracy of accumulated data, for example, editing and run-to-run totals. Option B data file control procedures ensure that only authorized processing occurs to stored data, for example, transaction logs. Option C output controls ensure that data delivered to users will be presented, formatted and delivered in a consistent and secure manner, for example, using report distribution. Option D "Application Controls" is a general term comprising all kinds of controls used in an application.
5. C is the correct answer. Integrated test facility compares processing output with independently calculated data. Explanation: An integrated test facility is considered a useful audit tool because it uses the same programs to compare processing using independently calculated data. This involves setting up dummy entities on an application system and processing test or production data against the entity as a means of verifying processing accuracy. Option A, B and D are not the dimensions of integrated test facility.

References

1. ISA2.0 Module 5
2. Selection of SDLC Models (References *Software Development Life Cycle Models and Methodologies*. (2012, 3). Retrieved from melsatar.blog: [Software Development Life Cycle Models and Methodologies](#))
3. CISA Review Manual 26th Edition.
4. CISA Review Manual 27th Edition.
5. <https://www.iso.org/standard/35733.html>
6. <https://www.iso.org/standard/35747.html>
7. <https://csrc.nist.gov/csrc/media/publications/conferencepaper/1996/10/22/proceeding-of-the-19th-nissc-1996/documents/paper001/article.pdf>

Annexure

A-1 ISO/IEC 25010:2011 defines:

<https://www.iso.org/standard/35733.html>

1. A quality in use model is composed of five characteristics (some of which are further sub-divided into sub-characteristics) that relate to the outcome of interaction when a product is used in a particular context. This system model is applicable to the complete human-computer system, including both computer systems in use and software products in use.
2. A product quality model is composed of eight characteristics (which are further subdivided into sub-characteristics) that relate to static properties of software and dynamic properties of the computer system. The model is applicable to both computer systems and software products.

The characteristics defined by both models are relevant to all software products and computer systems. The characteristics and sub-characteristics provide consistent terminology for specifying, measuring and evaluating system and software product quality. They also provide a set of quality characteristics against which stated quality requirements can be compared for completeness.

Although the scope of the product quality model is intended to be software and computer systems, many of the characteristics are also relevant to wider systems and services.

ISO 25010 has eight product quality characteristics (in contrast to ISO 9126's six), and 31 sub-characteristics.

- "Functionality" is renamed "Functional Suitability". "Functional Completeness" is added as a sub-characteristic, and "Interoperability" and "Security" are moved elsewhere. "Accuracy" is renamed "Functional Correctness", and "Suitability" is renamed "Functional Appropriateness".
- "Efficiency" is renamed "Performance Efficiency". "Capacity" is added as a sub-characteristic.
- "Compatibility" is a new characteristic, with "Co-existence" moved from "Portability" and "Interoperability" moved from "Functionality".
- "Usability" has new sub-characteristics of "user error protection" and "accessibility" (used in people with a wide range of characteristics). "Understandability" is renamed "Appropriateness Recognizability", and "Attractiveness" is renamed "User Interface Aesthetics".
- "Reliability" has a new sub-characteristic of "Availability" (when required for use).

- "Security" is a new characteristic with sub-characteristics of "Confidentiality" (data accessible only by those authorized), "Integrity" (protection from unauthorized modification), "Non-repudiation" (actions can be proven to have taken place), "Accountability" (actions can be traced to who did them), and "Authenticity" (identity can be proved to be the one claimed).
- "Maintainability" has new sub-characteristics of "Modularity" (changes in one component have a minimal impact on others) and "Reusability"; "Changeability" and "Stability" are rolled up into "Modifiability".
- "Portability" has "Co-existence" moved elsewhere.

ISO/IEC 25012:2008

<https://www.iso.org/standard/35733.html>

ISO/IEC 25012:2008 defines a general data quality model for data retained in a structured format within a computer system.

ISO/IEC 25012:2008 can be used to establish data quality requirements, define data quality measures, or plan and perform data quality evaluations. It could be used, for example,

- To define and evaluate data quality requirements in data production, acquisition and integration processes,
- To identify data quality assurance criteria, also useful for re-engineering, assessment and improvement of data,
- To evaluate the compliance of data with legislation and/or requirements.

ISO/IEC 25012:2008 categorizes quality attributes into fifteen characteristics considered by two points of view: inherent and system dependent. Data quality characteristics will be of varying importance and priority to different stakeholders.

ISO/IEC 25023:2016

<https://www.iso.org/standard/35747.html>

ISO/IEC 25023:2016 defines quality measures for quantitatively evaluating system and software product quality in terms of characteristics and sub-characteristics defined in ISO/IEC 25010 and is intended to be used together with ISO/IEC 25010.

ISO/IEC 25023:2016 contains the following:

- a basic set of quality measures for each characteristic and sub-characteristic;
- an explanation of how to apply software product and system quality measures.

The proposed quality measures are primarily intended to be used for quality assurance and improvement of system and software products during or post the development life cycle process.

The main users of ISO/IEC 25023:2016 are people carrying out quality requirement specification and evaluation activities as part of the following:

- Development: Including requirements analysis, design specification, coding and testing through acceptance during the life cycle process;
- Quality Management: Systematic examination of the software product or computer system, for example, when evaluating system or software product quality as part of quality assurance, quality control and quality certification;
- Supply: A contract with the acquirer for the supply of a system, software product or software service under the terms of a contract, for example, when validating quality at qualification test;
- Acquisition: Including product selection and acceptance testing, when acquiring or procuring a system, software product or software service from a supplier;
- Maintenance: Improvement of the software product or system based on quality measurement.